Attack on SHealS and HealS: the Second Wave of GPST

Yi-Fu Lai
joint work with Steven D. Galbraith
University of Auckland
New Zealand

September @ PQCrypto2022

Roadmap

1. What's this work about?
   - SIDH Key exchange
   - GPST Adaptive Attack [AC:GPST16]
   - A countermeasure for SIDH-type Schemes by Fouotsa and Petit [AC:FP21]
2. Quick Questions
3. Technical Overview
   - First Bit Extraction
   - Extraction of the maximal power of 2 divisor
   - Next Bit Extraction

Content

Content

Preliminaries

Quick Questions

Technical Overview

A Brief Intro/Setting for SIDH

- $p = 2^a 3^b - 1$ is a prime where $2^a \approx 3^b$.
- Elliptic curves: $E_A/\mathbb{F}_{p^2} : \; y^2 = x^3 + Ax^2 + x$.

A Brief Intro/Setting for SIDH

- $p = 2^a 3^b - 1$ is a prime where $2^a \approx 3^b$.
- Elliptic curves: $E_A/\mathbb{F}_{p^2}$: $y^2 = x^3 + Ax^2 + x$.
- An isogeny $\phi : E_A \to E_B$ is a morphism and also a group homomorphism, uniquely determined by the kernel and the image curve (up to isomorphism).
- For N not divisible by p,

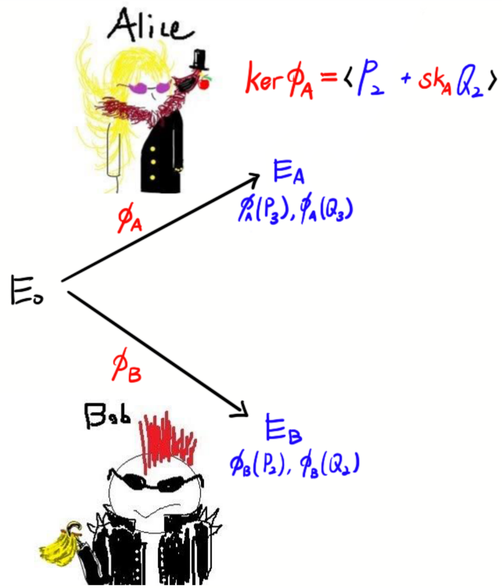
A Brief Intro/Setting for SIDH

- $p = 2^a 3^b - 1$ is a prime where $2^a \approx 3^b$.
- Elliptic curves: $E_A/\mathbb{F}_{p^2} : \; y^2 = x^3 + Ax^2 + x$.
- An isogeny $\phi : E_A \to E_B$ is a morphism and also a group homomorphism, uniquely determined by the kernel and the image curve (up to isomorphism).
- For N not divisible by p,

$$\begin{aligned} E[N] &= \{P \in E(\bar{\mathbb{F}}_p) \mid [N]P = \mathbf{O}\} \\ &\cong \mathbb{Z}_N \times \mathbb{Z}_N \end{aligned}$$

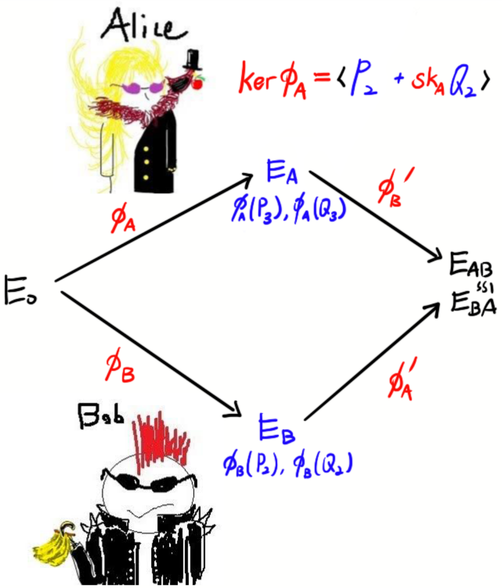
SIDH Key Exchange

- $E[2^a] \cong \mathbb{Z}_{2^a} \times \mathbb{Z}_{2^a}$ with a basis $\{P_2, Q_2\}$.
- $E[3^b] \cong \mathbb{Z}_{3^b} \times \mathbb{Z}_{3^b}$ with a basis $\{P_3, Q_3\}$.
- Alice: $\mathsf{sk}_A \in [2^a]$
- $\ker(\phi_A) = \langle P_2 + \mathsf{sk}_A Q_2 \rangle$
- Bob: $\mathsf{sk}_B \in [3^b]$
- $\ker(\phi_B) = \langle P_3 + \mathsf{sk}_B Q_3 \rangle$



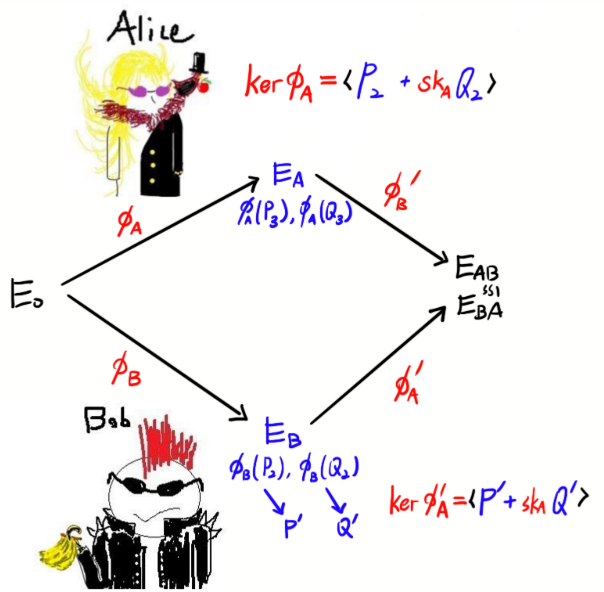
SIDH Key Exchange

- $E[2^a] \cong \mathbb{Z}_{2^a} \times \mathbb{Z}_{2^a}$ with a basis $\{P_2, Q_2\}$.
- $E[3^b] \cong \mathbb{Z}_{3^b} \times \mathbb{Z}_{3^b}$ with a basis $\{P_3, Q_3\}$.
- Alice: $\mathsf{sk}_A \in [2^a]$
- $\ker(\phi_A) = \langle P_2 + \mathsf{sk}_A Q_2 \rangle$
- Bob: $\mathsf{sk}_B \in [3^b]$
- $\ker(\phi_B) = \langle P_3 + \mathsf{sk}_B Q_3 \rangle$



SIDH Key Exchange

- $E[2^a] \cong \mathbb{Z}_{2^a} \times \mathbb{Z}_{2^a}$ with a basis $\{P_2, Q_2\}$.
- $E[3^b] \cong \mathbb{Z}_{3^b} \times \mathbb{Z}_{3^b}$ with a basis $\{P_3, Q_3\}$.
- Alice: $\mathsf{sk}_A \in [2^a]$
- $\ker(\phi_A) = \langle P_2 + \mathsf{sk}_A Q_2 \rangle$
- Bob: $\mathsf{sk}_B \in [3^b]$
- $\ker(\phi_B) = \langle P_3 + \mathsf{sk}_B Q_3 \rangle$



GPST Adaptive Attack

- (**Modeling**) Bob is the bad guy. Alice is an oracle on input $O_{\mathsf{sk}_A}(E_B, P', Q', E_{AB})$ and returns 1 iff

$$E_{AB} \cong E_B/\langle P' + \mathsf{sk}_A Q' \rangle,$$

$$e_{2^a}(P', Q') = e_{2^a}(P, Q)^{3^b}.$$

GPST Adaptive Attack

- (**Modeling**) Bob is the bad guy. Alice is an oracle on input $\mathcal{O}_{\mathsf{sk}_A}(E_B, P', Q', E_{AB})$ and returns 1 iff

$$E_{AB} \cong E_B/\langle P' + \mathsf{sk}_A Q' \rangle,$$

$$e_{2^a}(P', Q') = e_{2^a}(P, Q)^{3^b}.$$

- (**Assumption**) When $|G_1|, |G_2| \ll p$, with an overwhelming chance,

$$E_B/G_1 \cong E_B/G_2 \iff G_1 = G_2.$$

GPST Adaptive Attack

- (**Modeling**) Bob is the bad guy. Alice is an oracle on input $\mathcal{O}_{\mathsf{sk}_A}(E_B, P', Q', E_{AB})$ and returns 1 iff

$$E_{AB} \cong E_B/\langle P' + \mathsf{sk}_A Q' \rangle,$$

$$e_{2^a}(P', Q') = e_{2^a}(P, Q)^{3^b}.$$

- (**Assumption**) When $|G_1|, |G_2| \ll p$, with an overwhelming chance,

$$E_B/G_1 \cong E_B/G_2 \iff G_1 = G_2.$$

- Hence, on input $\mathcal{O}_{\mathsf{sk}_A}(E_B, P', Q', E_{AB})$, Alice returns 1 iff

$$\langle P + \mathsf{sk}_A Q \rangle = \langle P' + \mathsf{sk}_A Q' \rangle$$

$$e_{2^a}(P', Q') = e_{2^a}(P, Q)^{3^b}.$$

GPST Adaptive Attack

1. Bob honestly computes $E_B, P = \phi_B(P_2), Q = \phi_B(Q_2), E_{AB}$.
2. Let $P' = P$, $Q' = 2^{a-1}P + Q$. Then

$$\mathcal{O}_{\mathsf{sk}_A}(E_B, P', Q', E_{AB}) \to 1 \iff sk_A = 0 \mod 2.$$

⟨ Sketch of Pf ⟩: Firstly,

$$e_{2^a}(P', Q') = e_{2^a}(P, Q) = e_{2^a}(P, Q)^{3b}.$$

Claim

$$\langle P' + \mathsf{sk}_A Q' \rangle = \langle P + \mathsf{sk}_A Q \rangle \iff \mathsf{sk}_A\text{: even}$$

$$\begin{aligned}\langle P' + \mathsf{sk}_A Q' \rangle &= \langle P + \mathsf{sk}_A (2^{a-1}P + Q) \rangle \\ &= \langle P + \mathsf{sk}_A Q + \mathsf{sk}_A (2^{a-1}P) \rangle \\ &= \langle P + \mathsf{sk}_A Q \rangle \iff \mathsf{sk}_A\text{: even. } (2^a P = \mathbf{O})\end{aligned}$$

The Concept of GPST Attack

Take $a = 3$ for instance: $\langle P, Q\rangle = E[8] \cong \mathbb{Z}_8 \times \mathbb{Z}_8$

$\langle (\overset{P}{001}, \overset{Q}{000}) + (\overset{P}{000}, \overset{Q}{001})\mathsf{sk}_A \rangle$ (The correct kernel.)

The Concept of GPST Attack

Take $a = 3$ for instance: $\langle P, Q \rangle = E[8] \cong \mathbb{Z}_8 \times \mathbb{Z}_8$

$\langle (\overset{P}{001}, \overset{Q}{000}) + (\overset{P}{000}, \overset{Q}{001})\mathsf{sk}_A \rangle$ (The correct kernel.)

$\langle (001, 000) + (100, 001)\mathsf{sk}_A \rangle$ (The manipulated input.)

The Concept of GPST Attack

Take $a = 3$ for instance: $\langle P, Q\rangle = E[8] \cong \mathbb{Z}_8 \times \mathbb{Z}_8$

$$\langle(\overset{P}{001}, \overset{Q}{000}) + (\overset{P}{000}, \overset{Q}{001})\mathsf{sk}_A\rangle \quad \text{(The correct kernel.)}$$

$$\langle(001, 000) + (100, 001)\mathsf{sk}_A\rangle \quad \text{(The manipulated input.)}$$
$$=\langle(001, 000) + (000, 001)\mathsf{sk}_A + (100, 000)\mathsf{sk}_A\rangle$$

The Concept of GPST Attack

Take $a = 3$ for instance: $\langle P, Q \rangle = E[8] \cong \mathbb{Z}_8 \times \mathbb{Z}_8$

$$\langle (\overset{P}{001}, \overset{Q}{000}) + (\overset{P}{000}, \overset{Q}{001})\mathsf{sk}_A \rangle \quad \text{(The correct kernel.)}$$

$$\langle (001, 000) + (100, 001)\mathsf{sk}_A \rangle \quad \text{(The manipulated input.)}$$
$$= \langle (001, 000) + (000, 001)\mathsf{sk}_A + (100, 000)\mathsf{sk}_A \rangle$$

$\Rightarrow$ Get lsb sk_0.

The Concept of GPST Attack

Take $a = 3$ for instance: $\langle P, Q\rangle = E[8] \cong \mathbb{Z}_8 \times \mathbb{Z}_8$

$$\langle(\overset{P}{001}, \overset{Q}{000}) + (\overset{P}{000}, \overset{Q}{001})\mathsf{sk}_A\rangle \quad \text{(The correct kernel.)}$$

$$\langle(001, 000) + (100, 001)\mathsf{sk}_A\rangle \quad \text{(The manipulated input.)}$$
$$=\langle(001, 000) + (000, 001)\mathsf{sk}_A + (100, 000)\mathsf{sk}_A\rangle$$

$\Rightarrow$ Get lsb sk_0.

$$\langle(0\text{-}\mathsf{sk}_0 1, 000) + (010, 001)\mathsf{sk}_A\rangle \quad \text{(The manipulated input.)}$$

The Concept of GPST Attack

Take $a = 3$ for instance: $\langle P, Q\rangle = E[8] \cong \mathbb{Z}_8 \times \mathbb{Z}_8$

$$\langle (\overset{P}{001}, \overset{Q}{000}) + (\overset{P}{000}, \overset{Q}{001})\mathsf{sk}_A \rangle \quad \text{(The correct kernel.)}$$

$$\langle (001, 000) + (100, 001)\mathsf{sk}_A \rangle \quad \text{(The manipulated input.)}$$
$$= \langle (001, 000) + (000, 001)\mathsf{sk}_A + (100, 000)\mathsf{sk}_A \rangle$$

$\Rightarrow$ Get lsb sk_0.

$$\langle (0\text{-}\mathsf{sk}_0 1, 000) + (010, 001)\mathsf{sk}_A \rangle \quad \text{(The manipulated input.)}$$
$$= \langle (001, 000) + (000, 001)\mathsf{sk}_A + (010, 000)\mathsf{sk}_A + (0\text{-}\mathsf{sk}_0 0, 000) \rangle.$$

The Concept of GPST Attack

Take $a = 3$ for instance: $\langle P, Q\rangle = E[8] \cong \mathbb{Z}_8 \times \mathbb{Z}_8$

$$\langle (\overset{P}{001}, \overset{Q}{000}) + (\overset{P}{000}, \overset{Q}{001})\mathsf{sk}_A \rangle \quad \text{(The correct kernel.)}$$

$$\langle (001, 000) + (100, 001)\mathsf{sk}_A \rangle \quad \text{(The manipulated input.)}$$
$$= \langle (001, 000) + (000, 001)\mathsf{sk}_A + (100, 000)\mathsf{sk}_A \rangle$$

$\Rightarrow$ Get lsb sk_0.

$$\langle (0\text{-}\mathsf{sk}_0 1, 000) + (010, 001)\mathsf{sk}_A \rangle \quad \text{(The manipulated input.)}$$
$$= \langle (001, 000) + (000, 001)\mathsf{sk}_A + (010, 000)\mathsf{sk}_A + (0\text{-}\mathsf{sk}_0 0, 000) \rangle.$$
$$= \langle (001, 000) + (000, 001)\mathsf{sk}_A + (\mathsf{sk}_1\mathsf{sk}_0 0, 000) + (0\text{-}\mathsf{sk}_0 0, 000) \rangle.$$

The Concept of GPST Attack

Take $a = 3$ for instance: $\langle P, Q\rangle = E[8] \cong \mathbb{Z}_8 \times \mathbb{Z}_8$

$$\langle(\overset{P}{001}, \overset{Q}{000}) + (\overset{P}{000}, \overset{Q}{001})\mathsf{sk}_A\rangle \quad \text{(The correct kernel.)}$$

$$\langle(001, 000) + (100, 001)\mathsf{sk}_A\rangle \quad \text{(The manipulated input.)}$$
$$=\langle(001, 000) + (000, 001)\mathsf{sk}_A + (100, 000)\mathsf{sk}_A\rangle$$

$\Rightarrow$ Get lsb sk_0.

$$\langle(0\text{-}\mathsf{sk}_0 1, 000) + (010, 001)\mathsf{sk}_A\rangle \quad \text{(The manipulated input.)}$$
$$=\langle(001, 000) + (000, 001)\mathsf{sk}_A + (010, 000)\mathsf{sk}_A + (0\text{-}\mathsf{sk}_0 0, 000)\rangle.$$
$$=\langle(001, 000) + (000, 001)\mathsf{sk}_A + (\mathsf{sk}_1\mathsf{sk}_0 0, 000) + (0\text{-}\mathsf{sk}_0 0, 000)\rangle.$$
$$=\langle(001, 000) + (000, 001)\mathsf{sk}_A + (\mathsf{sk}_1 00, 000)\rangle.$$

The Concept of GPST Attack

Take $a = 3$ for instance: $\langle P, Q\rangle = E[8] \cong \mathbb{Z}_8 \times \mathbb{Z}_8$

$$\langle(\overset{P}{001}, \overset{Q}{000}) + (\overset{P}{000}, \overset{Q}{001})\mathsf{sk}_A\rangle$$ (The correct kernel.)

$\langle(001, 000) + (100, 001)\mathsf{sk}_A\rangle$ (The manipulated input.)

$=\langle(001, 000) + (000, 001)\mathsf{sk}_A + (100, 000)\mathsf{sk}_A\rangle$

$\Rightarrow$ Get lsb sk_0.

$\langle(0\text{-}\mathsf{sk}_0 1, 000) + (010, 001)\mathsf{sk}_A\rangle$ (The manipulated input.)

$=\langle(001, 000) + (000, 001)\mathsf{sk}_A + (010, 000)\mathsf{sk}_A + (0\text{-}\mathsf{sk}_0 0, 000)\rangle.$

$=\langle(001, 000) + (000, 001)\mathsf{sk}_A + (\mathsf{sk}_1\mathsf{sk}_0 0, 000) + (0\text{-}\mathsf{sk}_0 0, 000)\rangle.$

$=\langle(001, 000) + (000, 001)\mathsf{sk}_A + (\mathsf{sk}_1 00, 000)\rangle.$

$\Rightarrow$ Get the second lsb sk_1.

(Rmk: one has to scale the coefficient to have pass the pairing check.)

Is this Bad?

- This can be easily prevented by using the FO-transform-type method: Bob always uses an ephemeral secret key and reveal it to Alice.
  - This results in having static-ephemeral only cryptosystem.

Is this Bad?

- This can be easily prevented by using the FO-transform-type method: Bob always uses an ephemeral secret key and reveal it to Alice.
  - This results in having static-ephemeral only cryptosystem.

- Alternative: use either ZK proof systems or the multiple-public-keys techniques e.g.[UJ:20, SAC:AJL17].

Is this Bad?

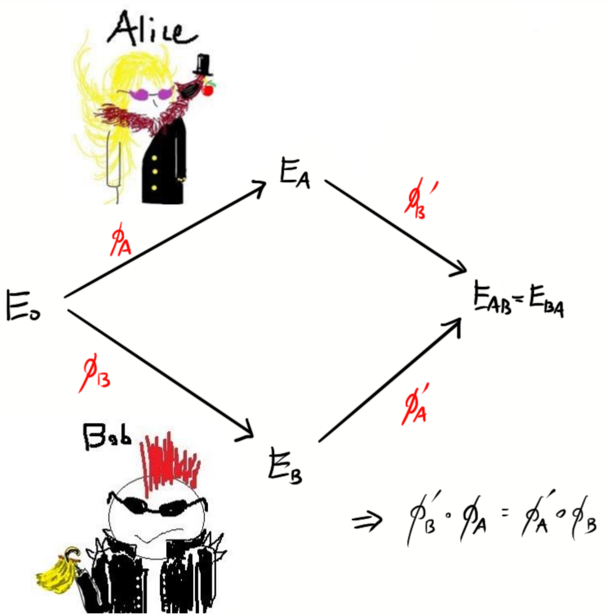
- This can be easily prevented by using the FO-transform-type method: Bob always uses an ephemeral secret key and reveal it to Alice.
  - This results in having static-ephemeral only cryptosystem.
- Alternative: use either ZK proof systems or the multiple-public-keys techniques e.g.[UJ:20, SAC:AJL17].
  - This results in the number of isogeny compuations non-constant in λ.

Is this Bad?

- This can be easily prevented by using the FO-transform-type method: Bob always uses an ephemeral secret key and reveal it to Alice.
  - This results in having static-ephemeral only cryptosystem.
- Alternative: use either ZK proof systems or the multiple-public-keys techniques e.g.[UJ:20, SAC:AJL17].
  - This results in the number of isogeny compuations non-constant in λ.
- [AC:FP21] gives an interactive proof system for the correctness of the public key.

A Proposed Countermeasure

- A countermeasure proposed by Fouotsa and Petit in [AC:FP21].
- The high-level idea is to use *commutativity* of isogenies [Leo20].



- If Bob manipulates the points in his public key, then the final evaluation will not match.

What Did We Do?

- We notice the flaw in the proof of the proof system in [AC:FP21].
- Based on the flaw, we derive a variant of GPST attack that adaptively recovers users' secret keys again.

What Did We Do?

- We notice the flaw in the proof of the proof system in [AC:FP21].
- Based on the flaw, we derive a variant of GPST attack that adaptively recovers users' secret keys again.
- The attack is as efficient and effective as the GPST attack.

Content

Quick Questions

- Can the Castryck-Decru (passive) attack (2022/975) apply to this scheme?
  - Yes, but not in polynomial-time theoretically by the current version (17 Sep 2022) due to the unknown endomorphism ring.

Quick Questions

- Can the Castryck-Decru (passive) attack (2022/975) apply to this scheme?
  - Yes, but not in polynomial-time theoretically by the current version (17 Sep 2022) due to the unknown endomorphism ring.
- How about the Robert (passive) attack (2022/1038)?
  - Yes, and in polynomial-time theoretically.

Quick Questions

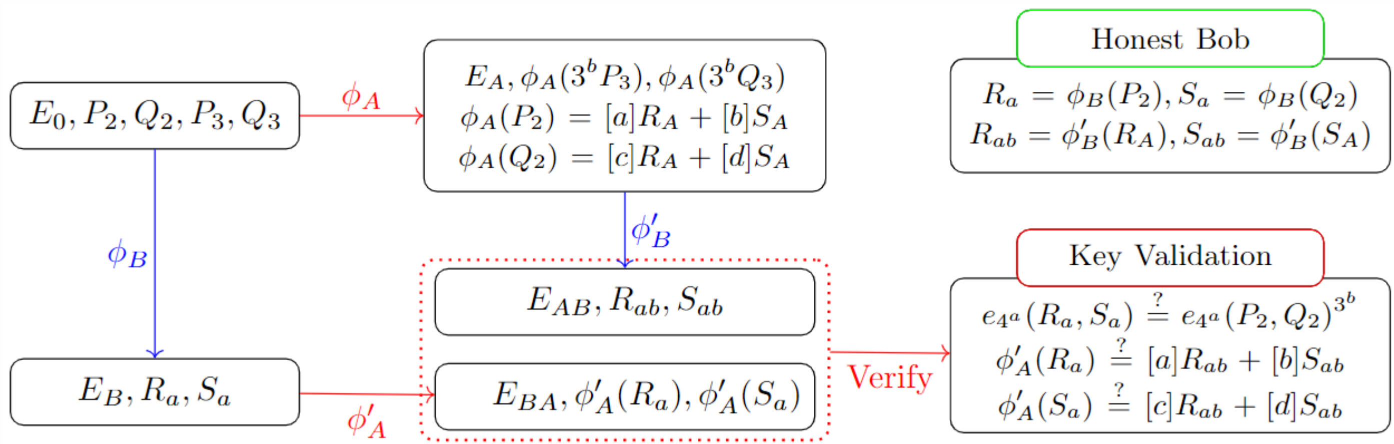
- Can the Castryck-Decru (passive) attack (2022/975) apply to this scheme?
  - Yes, but not in polynomial-time theoretically by the current version (17 Sep 2022) due to the unknown endomorphism ring.
- How about the Robert (passive) attack (2022/1038)?
  - Yes, and in polynomial-time theoretically.
- What's the salvage value of this attack?
  - No practical. Only theoretical values.

Content

HealSIDH and Its Key Validation Mechanism

- $\{P_2, Q_2\}$: basis for $E[2^{2a}]$
- Alice: $\mathsf{sk}_A \in [2^a]$
- $\ker(\phi_A) = \langle 2^a P_2 + \mathsf{sk}_A 2^a Q_2 \rangle$

- $\{P_3, Q_3\}$: basis for $E[3^{2b}]$
- Bob: $\mathsf{sk}_B \in [3^b]$
- $\ker(\phi_B) = \langle 3^b P_3 + \mathsf{sk}_B 3^b Q_3 \rangle$



Modeling

- Say Bob is the bad guy; Alice is the victim of the attack.
- Say Alice is an oracle on input $(E_B, R_a, S_a, R_{ab}, S_{ab})$ returning 1 iff the following three equations holds:

$$e_{4^a}(R_a, S_a) = e_{4^a}(P_2, Q_2)^{3^b}, \qquad \text{(Pairing Eq)}$$

$$\phi'_A(R_a) = [w]R_{ab} + [x]S_{ab} \in E_{BA}, \qquad \text{(Eq. 1)}$$

$$\phi'_A(S_a) = [y]R_{ab} + [z]S_{ab} \in E_{BA}, \qquad \text{(Eq. 2)}$$

where

$$\phi'_A : E_B \to E_{BA}$$

$$\ker(\phi'_A) = \langle [2^a]R_a + [\mathsf{sk}_A 2^a]S_a \rangle \subset E_B. \qquad \text{(Kernel Eq)}$$

Manipulate R_a, S_a

- Say Alice is an oracle on input $(E_B, R_a, S_a, R_{ab}, S_{ab})$ returning 1 iff the following three equations holds.
- We will only manipulate ... $(E_B, R_a, S_a, R_{ab}, S_{ab})$

$$e_{4^a}(R_a, S_a) = e_{4^a}(P_2, Q_2)^{3^b}, \qquad \text{(Pairing Eq)}$$

$$\phi'_A(R_a) = [w]R_{ab} + [x]S_{ab} \in E_{BA}, \qquad \text{(Eq. 1)}$$

$$\phi'_A(S_a) = [y]R_{ab} + [z]S_{ab} \in E_{BA}, \qquad \text{(Eq. 2)}$$

where

$$\phi'_A : E_B \to E_{BA}$$

$$\ker(\phi'_A) = \langle [2^a]R_a + [\mathsf{sk}_A 2^a]S_a \rangle \subset E_B. \qquad \text{(Kernel Eq)}$$

Lemmata

1. $\langle R_a, S_a \rangle = E_B[2^{2a}]$.

Lemmata

1. $\langle R_a, S_a \rangle = E_B[2^{2a}]$.
2. Recall that

$$\begin{aligned} \phi'_A(R_a) &= [w]R_{ab} + [x]S_{ab} \in E_{BA}, \\ \phi'_A(S_a) &= [y]R_{ab} + [z]S_{ab} \in E_{BA}, \end{aligned}$$

Lemmata

1. $\langle R_a, S_a \rangle = E_B[2^{2a}]$.
2. Recall that

$$
\begin{aligned}
\phi'_A(R_a) &= [w]R_{ab} + [x]S_{ab} \in E_{BA}, \\
\phi'_A(S_a) &= [y]R_{ab} + [z]S_{ab} \in E_{BA},
\end{aligned}
$$

we can prove that

$$w + \mathsf{sk}_A y = x + \mathsf{sk}_A z = 0 \mod 2^a$$

$(w, x, y, z \in [2^{2a}], \mathsf{sk}_A \in [2^a])$.

Lemmata

1. $\langle R_a, S_a \rangle = E_B[2^{2a}]$.
2. Recall that

$$\begin{aligned}\phi'_A(R_a) &= [w]R_{ab} + [x]S_{ab} \in E_{BA},\\ \phi'_A(S_a) &= [y]R_{ab} + [z]S_{ab} \in E_{BA},\end{aligned}$$

we can prove that

$$w + \mathsf{sk}_A y = x + \mathsf{sk}_A z = 0 \mod 2^a$$

$(w, x, y, z \in [2^{2a}], \mathsf{sk}_A \in [2^a])$.

$\Rightarrow$ Information of sk_A is hidden in the lower bits of w, x, y, z.

The First Bit Extraction

Recall: $\phi'_A \begin{pmatrix} R_a \\ S_a \end{pmatrix} = \begin{pmatrix} w & x \\ y & z \end{pmatrix} \begin{pmatrix} R_{ab} \\ S_{ab} \end{pmatrix}$

The First Bit Extraction

$$\textbf{Recall: } \phi'_A \begin{pmatrix} R_a \\ S_a \end{pmatrix} = \begin{pmatrix} w & x \\ y & z \end{pmatrix} \begin{pmatrix} R_{ab} \\ S_{ab} \end{pmatrix}$$

- Find special matrices $\mathbf{P}_1, \mathbf{P}_2$ s.t. $\mathbf{P}_1 \begin{pmatrix} w & x \\ y & z \end{pmatrix} = \begin{pmatrix} w & x \\ y & z \end{pmatrix} \mathbf{P}_2$ conditioned on parity of w, x, y, z.
- Also, $\det(\mathbf{P}_1) = 1$. (For the pairing eq.)

The First Bit Extraction

$$\textbf{Recall: } \phi'_A \begin{pmatrix} R_a \\ S_a \end{pmatrix} = \begin{pmatrix} w & x \\ y & z \end{pmatrix} \begin{pmatrix} R_{ab} \\ S_{ab} \end{pmatrix}$$

- Find special matrices $\mathbf{P}_1, \mathbf{P}_2$ s.t. $\mathbf{P}_1 \begin{pmatrix} w & x \\ y & z \end{pmatrix} = \begin{pmatrix} w & x \\ y & z \end{pmatrix} \mathbf{P}_2$ conditioned on parity of w, x, y, z.
- Also, $\det(\mathbf{P}_1) = 1$. (For the pairing eq.)
- With such a pair, invoking the oracle by $(E_B, R'_a, S'_a, R'_{ab}, S'_{ab})$ where

$$\begin{pmatrix} R'_a \\ S'_a \end{pmatrix} = \mathbf{P}_1 \begin{pmatrix} R_a \\ S_a \end{pmatrix}, \begin{pmatrix} R'_{ab} \\ S'_{ab} \end{pmatrix} = \mathbf{P}_2 \begin{pmatrix} R_{ab} \\ S_{ab} \end{pmatrix}.$$

- It returns 1 iff the the commutativity condition holds.

The First Bit Extraction

We take

$$\mathbf{P}_1 = \begin{pmatrix} 1 & 0 \\ 2^{2a-1} & 1 \end{pmatrix}, \mathbf{P}_2 = \mathbf{I}_2.$$

The commutativity holds iff $w = x = 0 \mod 2$.

The First Bit Extraction

We take

$$\mathbf{P}_1 = \begin{pmatrix} 1 & 0 \\ 2^{2a-1} & 1 \end{pmatrix}, \mathbf{P}_2 = \mathbf{I}_2.$$

The commutativity holds iff $w = x = 0 \mod 2$.

Recall $w + \mathsf{sk}_A y = x + \mathsf{sk}_A z = 0 \mod 2^a$ $(w, x, y, z \in [2^{2a}], \mathsf{sk}_A \in [2^a])$.

- We can prove that y, z cannot be both even.
- The commutativity holds iff $\mathsf{sk}_A = 0 \mod 2$.
- The first bit of $\mathsf{sk}_A = 0$ if and only if the oracle returns 1.
- The lsb of sk_A is extracted!

Recovering Higher Bits (High-level Idea)

Base on $w + \mathsf{sk}_A y = x + \mathsf{sk}_A z = 0 \mod 2^a$, we can write

$$\phi'_A \begin{pmatrix} R_a \\ S_a \end{pmatrix} = \begin{pmatrix} -\mathsf{sk}_A y \mod 2^a + * & -\mathsf{sk}_A z \mod 2^a + * \\ y & z \end{pmatrix} \begin{pmatrix} R_{ab} \\ S_{ab} \end{pmatrix}$$

Recovering Higher Bits (High-level Idea)

Base on $w + \mathsf{sk}_A y = x + \mathsf{sk}_A z = 0 \mod 2^a$, we can write

$$\phi'_A \begin{pmatrix} R_a \\ S_a \end{pmatrix} = \begin{pmatrix} -\mathsf{sk}_A y \mod 2^a + * & -\mathsf{sk}_A z \mod 2^a + * \\ y & z \end{pmatrix} \begin{pmatrix} R_{ab} \\ S_{ab} \end{pmatrix}$$

- Use the homomorphism ϕ'_A to launch GPST-type attack:

$$R'_a = [1 + 2^{2a-2}]R_a + [\mathsf{sk}_0 2^{2a-2}]S_a,$$

  where sk_0 is $\mathsf{sk}_A \mod 2$, just extracted.
- (Eq1) $\phi'_A(R'_a) = \phi'_A(R_a) \iff \mathsf{sk}_1 = 0.$

Recovering Higher Bits (High-level Idea)

Base on $w + \mathsf{sk}_A y = x + \mathsf{sk}_A z = 0 \mod 2^a$, we can write

$$\phi'_A \begin{pmatrix} R_a \\ S_a \end{pmatrix} = \begin{pmatrix} -\mathsf{sk}_A y \mod 2^a + * & -\mathsf{sk}_A z \mod 2^a + * \\ y & z \end{pmatrix} \begin{pmatrix} R_{ab} \\ S_{ab} \end{pmatrix}$$

- Use the homomorphism ϕ'_A to launch GPST-type attack:

$$R'_a = [1 + 2^{2a-2}]R_a + [\mathsf{sk}_0 2^{2a-2}]S_a,$$

  where sk_0 is $\mathsf{sk}_A \mod 2$, just extracted.
- (Eq1) $\phi'_A(R'_a) = \phi'_A(R_a) \iff \mathsf{sk}_1 = 0$.
- (Eq2) $\phi'_A(S_a) = \phi'_A(S_a)$ always.

Recovering Higher Bits (High-level Idea)

Base on $w + \mathsf{sk}_A y = x + \mathsf{sk}_A z = 0 \mod 2^a$, we can write

$$\phi'_A \begin{pmatrix} R_a \\ S_a \end{pmatrix} = \begin{pmatrix} -\mathsf{sk}_A y \mod 2^a + * & -\mathsf{sk}_A z \mod 2^a + * \\ y & z \end{pmatrix} \begin{pmatrix} R_{ab} \\ S_{ab} \end{pmatrix}$$

- Use the homomorphism ϕ'_A to launch GPST-type attack:

$$R'_a = [1 + 2^{2a-2}]R_a + [\mathsf{sk}_0 2^{2a-2}]S_a,$$

  where sk_0 is $\mathsf{sk}_A \mod 2$, just extracted.
- (Eq1) $\phi'_A(R'_a) = \phi'_A(R_a) \iff \mathsf{sk}_1 = 0$.
- (Eq2) $\phi'_A(S_a) = \phi'_A(S_a)$ always.
- (KernelEq) $\ker(\phi'_A) = \langle [2^a]R_a + [\mathsf{sk}_A 2^a]S_a \rangle = \langle [2^a]R'_a + [\mathsf{sk}_A 2^a]S_a \rangle$ ☺

Recovering Higher Bits (High-level Idea)

Base on $w + \mathsf{sk}_A y = x + \mathsf{sk}_A z = 0 \mod 2^a$, we can write

$$\phi'_A \begin{pmatrix} R_a \\ S_a \end{pmatrix} = \begin{pmatrix} -\mathsf{sk}_A y \mod 2^a + * & -\mathsf{sk}_A z \mod 2^a + * \\ y & z \end{pmatrix} \begin{pmatrix} R_{ab} \\ S_{ab} \end{pmatrix}$$

- Use the homomorphism ϕ'_A to launch GPST-type attack:

$$R'_a = [1 + 2^{2a-2}]R_a + [\mathsf{sk}_0 2^{2a-2}]S_a,$$

  where sk_0 is $\mathsf{sk}_A \mod 2$, just extracted.
- (Eq1) $\phi'_A(R'_a) = \phi'_A(R_a) \iff \mathsf{sk}_1 = 0$.
- (Eq2) $\phi'_A(S_a) = \phi'_A(S_a)$ always.
- (KernelEq) $\ker(\phi'_A) = \langle [2^a]R_a + [\mathsf{sk}_A 2^a]S_a \rangle = \langle [2^a]R'_a + [\mathsf{sk}_A 2^a]S_a \rangle$ ☺
- (PairingEq) But $e_{4^a}(R'_a, S_a) \neq e_{4^a}(R_a, S_a) = e_{4^a}(P_2, Q_2)^{3^b}$ ☹ (The scaling method won't work due to the Eq 2 & Kernel Eq.)

Recovering Higher Bits (High-level Idea)

Base on $w + \mathsf{sk}_A y = x + \mathsf{sk}_A z = 0 \mod 2^a$, we can write

$$\phi'_A \begin{pmatrix} R_a \\ S_a \end{pmatrix} = \begin{pmatrix} -\mathsf{sk}_A y \mod 2^a + * & -\mathsf{sk}_A z \mod 2^a + * \\ y & z \end{pmatrix} \begin{pmatrix} R_{ab} \\ S_{ab} \end{pmatrix}$$

- Use the homomorphism ϕ'_A to launch GPST-type attack:

$$R'_a = [1 + 2^{2a-2}]R_a + [\mathsf{sk}_0 2^{2a-2}]S_a,$$

  where sk_0 is $\mathsf{sk}_A \mod 2$, just extracted.
- (Eq1) $\phi'_A(R'_a) = \phi'_A(R_a) \iff \mathsf{sk}_1 = 0.$
- (Eq2) $\phi'_A(S_a) = \phi'_A(S_a)$ always.
- (KernelEq) $\ker(\phi'_A) = \langle [2^a]R_a + [\mathsf{sk}_A 2^a]S_a \rangle = \langle [2^a]R'_a + [\mathsf{sk}_A 2^a]S_a \rangle$ ☺
- (PairingEq) But $e_{4^a}(R'_a, S_a) \neq e_{4^a}(R_a, S_a) = e_{4^a}(P_2, Q_2)^{3^b}$ ☹ (The scaling method won't work due to the Eq 2 & Kernel Eq.)
- The oracle taking as input $(E_B, R'_a, S_a, R_{ab}, S_{ab})$ will return 0. ☹

Recovering the Higher Bits (High-level Idea)

$$\phi'_A \begin{pmatrix} R_a \\ S_a \end{pmatrix} = \begin{pmatrix} w & x \\ -\mathsf{sk}_A^{-1} w \mod 2^a + * & -\mathsf{sk}_A^{-1} x \mod 2^a + * \end{pmatrix} \begin{pmatrix} R_{ab} \\ S_{ab} \end{pmatrix}$$

Assume sk_A is invertible modulo 2^a, then

$$-w\mathsf{sk}_A^{-1} = y, -x\mathsf{sk}_A^{-1} = z \mod 2^a$$

Recovering the Higher Bits (High-level Idea)

$$\phi'_A \begin{pmatrix} R_a \\ S_a \end{pmatrix} = \begin{pmatrix} & w & & x & \\ -\mathsf{sk}_A^{-1} w & \mod 2^a + * & -\mathsf{sk}_A^{-1} x & \mod 2^a + * \end{pmatrix} \begin{pmatrix} R_{ab} \\ S_{ab} \end{pmatrix}$$

Assume sk_A is invertible modulo 2^a, then

$$-w\mathsf{sk}_A^{-1} = y, -x\mathsf{sk}_A^{-1} = z \mod 2^a$$

- $R'_a = [1 + 2^{2a-2}]R_a + [\mathsf{sk}_0 2^{2a-2}]S_a,$
- $S'_a = [\mathsf{sk}_0^{-1} 2^{2a-2}]R_a + [1 - 2^{2a-2}]S_a,$

Recovering the Higher Bits (High-level Idea)

$$\phi'_A\begin{pmatrix}R_a\\ S_a\end{pmatrix}=\begin{pmatrix} & w & & x & \\ -\mathsf{sk}_A^{-1}w & \bmod 2^a+* & -\mathsf{sk}_A^{-1}x & \bmod 2^a+* \end{pmatrix}\begin{pmatrix}R_{ab}\\ S_{ab}\end{pmatrix}$$

Assume sk_A is invertible modulo 2^a, then

$$-w\mathsf{sk}_A^{-1}=y,-x\mathsf{sk}_A^{-1}=z \mod 2^a$$

- $R'_a=[1+2^{2a-2}]R_a+[\mathsf{sk}_0 2^{2a-2}]S_a,$
- $S'_a=[\mathsf{sk}_0^{-1}2^{2a-2}]R_a+[1-2^{2a-2}]S_a,$
- (Eq1) $\phi'_A(R'_a)=\phi'_A(R_a) \iff \mathsf{sk}_1=0.$

Recovering the Higher Bits (High-level Idea)

$$\phi'_A \begin{pmatrix} R_a \\ S_a \end{pmatrix} = \begin{pmatrix} w & x \\ -\mathsf{sk}_A^{-1} w \mod 2^a + * & -\mathsf{sk}_A^{-1} x \mod 2^a + * \end{pmatrix} \begin{pmatrix} R_{ab} \\ S_{ab} \end{pmatrix}$$

Assume sk_A is invertible modulo 2^a, then

$$-w\mathsf{sk}_A^{-1} = y, -x\mathsf{sk}_A^{-1} = z \mod 2^a$$

- $R'_a = [1 + 2^{2a-2}]R_a + [\mathsf{sk}_0 2^{2a-2}]S_a,$
- $S'_a = [\mathsf{sk}_0^{-1} 2^{2a-2}]R_a + [1 - 2^{2a-2}]S_a,$
- (Eq1) $\phi'_A(R'_a) = \phi'_A(R_a) \iff \mathsf{sk}_1 = 0.$
- (Eq2) $\phi'_A(S'_a) = \phi'_A(S_a) \iff (\mathsf{sk}_A)_1^{-1} = 0 \iff \mathsf{sk}_1 = 0.$

Recovering the Higher Bits (High-level Idea)

$$\phi'_A \begin{pmatrix} R_a \\ S_a \end{pmatrix} = \begin{pmatrix} & w & & x & \\ -\mathsf{sk}_A^{-1} w & \mod 2^a + * & -\mathsf{sk}_A^{-1} x & \mod 2^a + * \end{pmatrix} \begin{pmatrix} R_{ab} \\ S_{ab} \end{pmatrix}$$

Assume sk_A is invertible modulo 2^a, then

$$-w\mathsf{sk}_A^{-1} = y, -x\mathsf{sk}_A^{-1} = z \mod 2^a$$

- $R'_a = [1 + 2^{2a-2}]R_a + [\mathsf{sk}_0 2^{2a-2}]S_a$,
- $S'_a = [\mathsf{sk}_0^{-1} 2^{2a-2}]R_a + [1 - 2^{2a-2}]S_a$,
- (Eq1) $\phi'_A(R'_a) = \phi'_A(R_a) \iff \mathsf{sk}_1 = 0$.
- (Eq2) $\phi'_A(S'_a) = \phi'_A(S_a) \iff (\mathsf{sk}_A)_1^{-1} = 0 \iff \mathsf{sk}_1 = 0$.
- (PairingEq) And $e_{4^a}(R'_a, S'_a) = e_{4^a}(P_2, Q_2)^{3^b}$. ☺

Recovering the Higher Bits (High-level Idea)

$$\phi'_A \begin{pmatrix} R_a \\ S_a \end{pmatrix} = \begin{pmatrix} w & x \\ -\mathsf{sk}_A^{-1} w \mod 2^a + * & -\mathsf{sk}_A^{-1} x \mod 2^a + * \end{pmatrix} \begin{pmatrix} R_{ab} \\ S_{ab} \end{pmatrix}$$

Assume sk_A is invertible modulo 2^a, then

$$-w\mathsf{sk}_A^{-1} = y, -x\mathsf{sk}_A^{-1} = z \mod 2^a$$

- $R'_a = [1 + 2^{2a-2}]R_a + [\mathsf{sk}_0 2^{2a-2}]S_a,$
- $S'_a = [\mathsf{sk}_0^{-1} 2^{2a-2}]R_a + [1 - 2^{2a-2}]S_a,$
- (Eq1) $\phi'_A(R'_a) = \phi'_A(R_a) \iff \mathsf{sk}_1 = 0.$
- (Eq2) $\phi'_A(S'_a) = \phi'_A(S_a) \iff (\mathsf{sk}_A)_1^{-1} = 0 \iff \mathsf{sk}_1 = 0.$
- (PairingEq) And $e_{4^a}(R'_a, S'_a) = e_{4^a}(P_2, Q_2)^{3^b}$. ☺
- (KernelEq) Also
  $\ker(\phi'_A) = \langle [2^a]R_a + [\mathsf{sk}_A 2^a]S_a \rangle = \langle [2^a]R'_a + [\mathsf{sk}_A 2^a]S'_a \rangle$ ☺

Recovering the Higher Bits (High-level Idea)

$$\phi'_A \begin{pmatrix} R_a \\ S_a \end{pmatrix} = \begin{pmatrix} w & x \\ -\mathsf{sk}_A^{-1} w \mod 2^a + * & -\mathsf{sk}_A^{-1} x \mod 2^a + * \end{pmatrix} \begin{pmatrix} R_{ab} \\ S_{ab} \end{pmatrix}$$

Assume sk_A is invertible modulo 2^a, then

$$-w\mathsf{sk}_A^{-1} = y, -x\mathsf{sk}_A^{-1} = z \mod 2^a$$

- $R'_a = [1 + 2^{2a-2}]R_a + [\mathsf{sk}_0 2^{2a-2}]S_a$,
- $S'_a = [\mathsf{sk}_0^{-1} 2^{2a-2}]R_a + [1 - 2^{2a-2}]S_a$,
- (Eq1) $\phi'_A(R'_a) = \phi'_A(R_a) \iff \mathsf{sk}_1 = 0$.
- (Eq2) $\phi'_A(S'_a) = \phi'_A(S_a) \iff (\mathsf{sk}_A)_1^{-1} = 0 \iff \mathsf{sk}_1 = 0$.
- (PairingEq) And $e_{4^a}(R'_a, S'_a) = e_{4^a}(P_2, Q_2)^{3^b}$. ☺
- (KernelEq) Also
  $\ker(\phi'_A) = \langle [2^a]R_a + [\mathsf{sk}_A 2^a]S_a \rangle = \langle [2^a]R'_a + [\mathsf{sk}_A 2^a]S'_a \rangle$ ☺

- What if sk_A is not invertible??

sk_A is Even.

Idea: Reuse the $\mathbf{P}_1, \mathbf{P}_2$ commutativity method, we can keep extracting the next bit until 1 appears.

- $R'_a = [1 + 2^{2a-1}]R_a$,
- $S'_a = [2^{2a-2}]R_a + [1 - 2^{2a-1}]S_a$,

sk_A is Even.

Idea: Reuse the $\mathbf{P}_1, \mathbf{P}_2$ commutativity method, we can keep extracting the next bit until 1 appears.

- $R'_a = [1 + 2^{2a-1}]R_a$,
- $S'_a = [2^{2a-2}]R_a + [1 - 2^{2a-1}]S_a$,
- (Eq1) $\phi'_A(R'_a) = \phi'_A(R_a)$: always. ☺
- (Eq2)

$$\phi'_A(S'_a) = \phi'_A(S_a) \iff \mathsf{sk}_1 2^{2a-1} - 2^{2a-1} = 0 \mod 2^{2a}$$
$$\iff \mathsf{sk}_1 = 1.$$

- (PairingEq) And $e_{4^a}(R'_a, S'_a) = e_{4^a}(P_2, Q_2)^{3^b}$. ☺
- (KernelEq) Also $\ker(\phi'_A) = \langle [2^a]R_a + [\mathsf{sk}_A 2^a]S_a \rangle = \langle [2^a]R'_a + [\mathsf{sk}_A 2^a]S'_a \rangle$ ☺

sk_A is Even.

Idea: Reuse the $\mathbf{P}_1, \mathbf{P}_2$ commutativity method, we can keep extracting the next bit until 1 appears.

- $R'_a = [1 + 2^{2a-1}]R_a$,
- $S'_a = [2^{2a-2}]R_a + [1 - 2^{2a-1}]S_a$,
- (Eq1) $\phi'_A(R'_a) = \phi'_A(R_a)$: always. ☺
- (Eq2)

$$\phi'_A(S'_a) = \phi'_A(S_a) \iff \mathsf{sk}_1 2^{2a-1} - 2^{2a-1} = 0 \mod 2^{2a}$$
$$\iff \mathsf{sk}_1 = 1.$$

- (PairingEq) And $e_{4^a}(R'_a, S'_a) = e_{4^a}(P_2, Q_2)^{3^b}$. ☺
- (KernelEq) Also $\ker(\phi'_A) = \langle [2^a]R_a + [\mathsf{sk}_A 2^a]S_a \rangle = \langle [2^a]R'_a + [\mathsf{sk}_A 2^a]S'_a \rangle$ ☺
- One can recursively use this approach to extract the maximal power of 2 in sk_A.

Extracting the Next Bit When sk_A is Even.

$$\phi'_A \begin{pmatrix} R_a \\ S_a \end{pmatrix} = \begin{pmatrix} -\mathsf{sk}_A y \mod 2^a + * & -\mathsf{sk}_A z \mod 2^a + * \\ y & z \end{pmatrix} \begin{pmatrix} R_{ab} \\ S_{ab} \end{pmatrix}$$

Say 2^j is the maximal power of 2 dividing sk_A and i lsbs of sk_A has been recovered, denoted by sk_ℓ.

Extracting the Next Bit When sk_A is Even.

$$\phi'_A \begin{pmatrix} R_a \\ S_a \end{pmatrix} = \begin{pmatrix} -\mathsf{sk}_A y \mod 2^a + * & -\mathsf{sk}_A z \mod 2^a + * \\ y & z \end{pmatrix} \begin{pmatrix} R_{ab} \\ S_{ab} \end{pmatrix}$$

Say 2^j is the maximal power of 2 dividing sk_A and i lsbs of sk_A has been recovered, denoted by sk_ℓ.

- Making queries on $(E_B, R'_a, S'_a, R_{ab}, S_{ab})$, where
- $R'_a = [1 + 2^{2a-i-1}2^j]R_a - [\mathsf{sk}_\ell 2^{2a-i-1}2^j]S_a,$
- $S'_a = [\widetilde{\mathsf{sk}_\ell} 2^{2a-i-1}]R_a + [1 + 2^{2a-i-1}2^j]S_a,$

($\widetilde{\mathsf{sk}_\ell}$ is the inverse of $\mathsf{sk}_\ell / 2^j \mod 2^i$)

Extracting the Next Bit When sk_A is Even.

$$\phi'_A\begin{pmatrix}R_a\\ S_a\end{pmatrix}=\begin{pmatrix}-\mathsf{sk}_A y \mod 2^a+* & -\mathsf{sk}_A z \mod 2^a+*\\ y & z\end{pmatrix}\begin{pmatrix}R_{ab}\\ S_{ab}\end{pmatrix}$$

Say 2^j is the maximal power of 2 dividing sk_A and i lsbs of sk_A has been recovered, denoted by sk_ℓ.

- Making queries on $(E_B, R'_a, S'_a, R_{ab}, S_{ab})$, where
- $R'_a = [1+2^{2a-i-1}2^j]R_a - [\mathsf{sk}_\ell 2^{2a-i-1}2^j]S_a$,
- $S'_a = [\widetilde{\mathsf{sk}_\ell}2^{2a-i-1}]R_a + [1+2^{2a-i-1}2^j]S_a$,

  ($\widetilde{\mathsf{sk}_\ell}$ is the inverse of $\mathsf{sk}_\ell/2^j \mod 2^i$)

- Paring/ Ker Eqs will hold.
- It returns 1 if the next bit is 0.

Extracting the Next Bit When sk_A is Even.

$$\phi'_A\begin{pmatrix}R_a\\ S_a\end{pmatrix}=\begin{pmatrix}-\mathsf{sk}_A y \mod 2^a + * & -\mathsf{sk}_A z \mod 2^a + *\\ y & z\end{pmatrix}\begin{pmatrix}R_{ab}\\ S_{ab}\end{pmatrix}$$

Say 2^j is the maximal power of 2 dividing sk_A and i lsbs of sk_A has been recovered, denoted by sk_ℓ.

- Making queries on $(E_B, R'_a, S'_a, R_{ab}, S_{ab})$, where
- $R'_a = [1 + 2^{2a-i-1}2^j]R_a - [\mathsf{sk}_\ell 2^{2a-i-1}2^j]S_a,$
- $S'_a = [\widetilde{\mathsf{sk}_\ell} 2^{2a-i-1}]R_a + [1 + 2^{2a-i-1}2^j]S_a,$

  ($\widetilde{\mathsf{sk}_\ell}$ is the inverse of $\mathsf{sk}_\ell/2^j \mod 2^i$)

- Paring/ Ker Eqs will hold.
- It returns 1 if the next bit is 0.
- Nicely done! ☺☺☺☺☺☺☺☺☺

We also generalize the result to any small primes and a more general form of the private keys.

Summary and Open Problems

Summary

- We present a new adaptive attack against SIDH-type schemes using the commutativity of isogenies.
- The adaptive attack runs in polynomial time.

Open Problems

- Is it possible to have an efficient variant of SIDH secure against the Castryck-Decru and Robert attacks? (e.g. 2022/1019,1054?)
- If so, can we have an efficient proof system to prevent the attack?

Thanks for listening!