# Cryptanalysis of an Oblivious PRF from Supersingular Isogenies

Andrea Basso, Péter Kutas, Simon-Philipp Merz, Christophe Petit and Antonio Sanso


March 2022

**ISG Seminar**

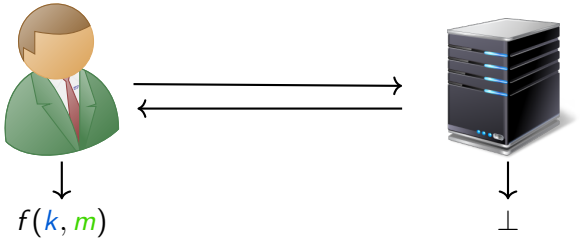
# Content

- Definition of (V)OPRFs
- Applications
  - OPAQUE
  - PrivacyPass
- Isogenies and SIDH
- OPRF from isogenies
- Cryptanalytic results
  - Polytime and subexponential attacks
  - Requirement for trusted setup

# Oblivious Pseudorandom Function (OPRF)

An OPRF is a two-party protocol to evaluate a PRF $f(k, m)$ where:

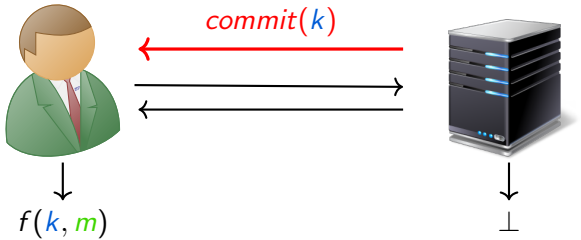
- The client learns $f(k, m)$, one evaluation of a PRF on a chosen input
- The server learns nothing about $m$



# Oblivious Pseudorandom Function (OPRF)

An OPRF is a two-party protocol to evaluate a PRF $f(k, m)$ where:

- The client learns $f(k, m)$, one evaluation of a PRF on a chosen input
- The server learns nothing about $m$



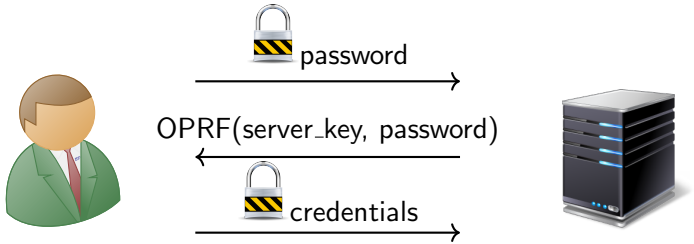

- An OPRF is called *verifable*, if the server proves to the client that output was computed using the key $k$

# OPAQUE: OPRF + PAKE

- Use passwords that never leave your device

How to check a password that you have never seen?

Registration Phase:

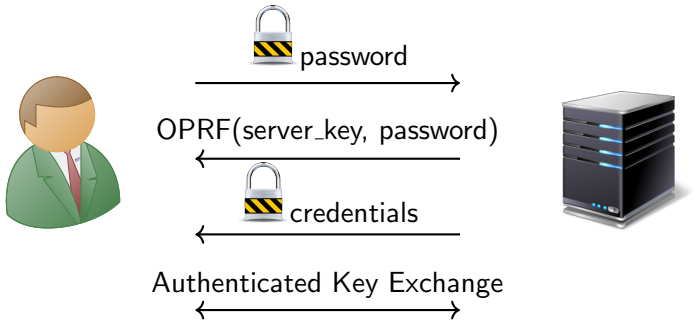


# OPAQUE: OPRF + PAKE

- Use passwords that never leave your device

How to check a password that you have never seen?

Login Phase:



# PrivacyPass

- Generate cryptographically 'blinded' tokens that can be signed by server after client authenticates themselves (e.g. CAPTCHA solution)
- Security properties:
  1. Unlinkability
  2. Unforgeability
- Construction:
  - VOPRF for issuance of tokens during blind signing phase
  - Verification of anonymous tokens during redemption phase

# Existing Constructions

Parameters: group $\mathbb{G}$ of order $q$, hash functions $H_1$, $H_2$ onto $\mathbb{G}$ and $\{0,1\}^\ell$ resp.

| Client $C(m)$ | | Server $S(k)$ |
|---|---|---|
| Pick $r \leftarrow_R \mathbb{Z}_q$ | | |
| Set $a \leftarrow (H_1(m))^r$ | $\xrightarrow{a}$ | |
| | | If $a \in \mathbb{G}$, set $b \leftarrow a^k$ |
| | $\xleftarrow{b}$ | |
| If $b \in \mathbb{G}$, set $v \leftarrow b^{1/r}$ | | |
| Output $H_2(m, v)$ | | |

# Existing Constructions

Parameters: group $\mathbb{G}$ of order $q$, hash functions $H_1$, $H_2$ onto $\mathbb{G}$ and $\{0,1\}^\ell$ resp.

| Client $C(m)$ | | Server $S(k)$ |
|---|---|---|
| Pick $r \leftarrow_R \mathbb{Z}_q$ | | |
| Set $a \leftarrow (H_1(m))^r$ | $\xrightarrow{a}$ | |
| | | If $a \in \mathbb{G}$, set $b \leftarrow a^k$ |
| | $\xleftarrow{b}$ | |
| If $b \in \mathbb{G}$, set $v \leftarrow b^{1/r}$ | | |
| Output $H_2(m, v)$ | | |

Post-quantum OPRF:

- Construction from lattices [ADDS19]
- Construction from isogenies [BKW20]

# Isogenies

Let $E_0$, $E_1$ be elliptic curves defined over a field $\overline{\mathbb{F}}_p$

- An *Isogeny* is non-constant rational map $\varphi : E_0 \to E_1$ that is also a group homomorphism
- The kernel of an isogeny determines the image curve up to isomorphism ($E_0/\ker(\varphi) := E_1$)
- Two curves $E_0$, $E_1$ are isomorphic if and only if they have the same $j$-invariant
- (Separable) isogenies correspond to subgroups of $E_0$ (order of subgroup equals degree of isogeny)

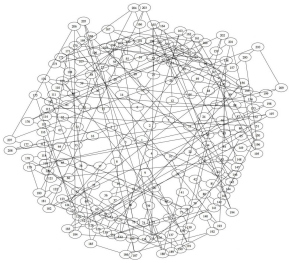
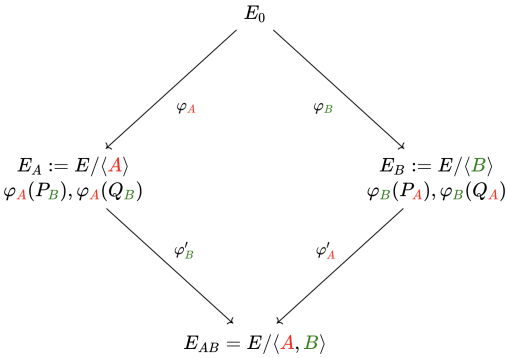


Figure: Image by D. Charles

# SIDH [JD11]

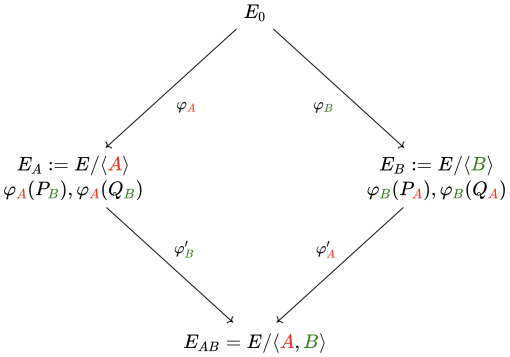
- Fix a prime $p$ such that $p = N_1 N_2 - 1$, $E_0/\mathbb{F}_p^2$ and bases $\langle P_A, Q_A \rangle = E_0[N_1]$, $\langle P_B, Q_B \rangle = E_0[N_2]$



# SIDH [JD11]

- Fix a prime $p$ such that $p = N_1 N_2 - 1$, $E_0/\mathbb{F}_p^2$ and bases $\langle P_A, Q_A \rangle = E_0[N_1]$, $\langle P_B, Q_B \rangle = E_0[N_2]$

- Alice's secret is $A := P_A + [\mathsf{sk}_A]Q_A$
- Bob's secret is $B := P_B + [\mathsf{sk}_B]Q_B$

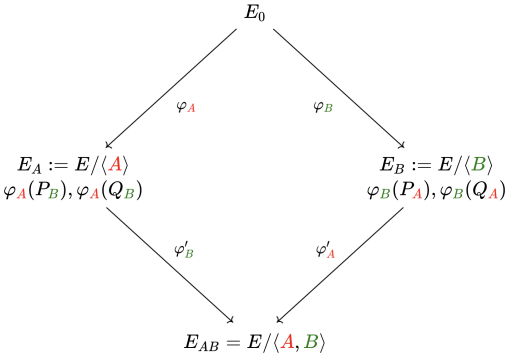


# SIDH [JD11]

- Fix a prime $p$ such that $p = N_1 N_2 - 1$, $E_0/\mathbb{F}_p^2$ and bases $\langle P_A, Q_A \rangle = E_0[N_1]$, $\langle P_B, Q_B \rangle = E_0[N_2]$

- Alice's secret is $A := P_A + [\mathsf{sk}_A]Q_A$
- Bob's secret is $B := P_B + [\mathsf{sk}_B]Q_B$

- Alice sends $E_A$, $\phi_A(P_B)$, $\phi_A(Q_B)$
- Bob sends $E_B$, $\phi_B(P_A)$, $\phi_B(Q_A)$

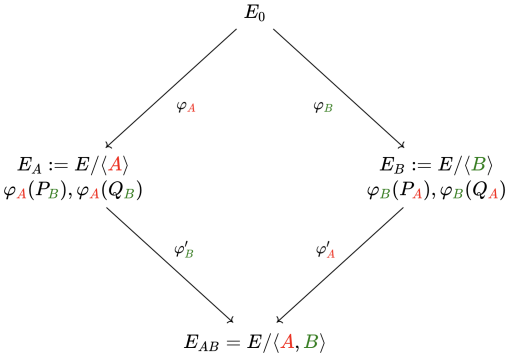


# SIDH [JD11]

- Fix a prime $p$ such that $p = N_1 N_2 - 1$, $E_0/\mathbb{F}_p^2$ and bases $\langle P_A, Q_A \rangle = E_0[N_1]$, $\langle P_B, Q_B \rangle = E_0[N_2]$

- Alice's secret is $A := P_A + [\mathsf{sk}_A]Q_A$
- Bob's secret is $B := P_B + [\mathsf{sk}_B]Q_B$

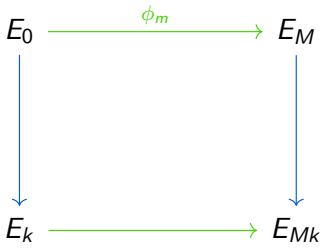
- Alice sends $E_A$, $\phi_A(P_B)$, $\phi_A(Q_B)$
- Bob sends $E_B$, $\phi_B(P_A)$, $\phi_B(Q_A)$




- The shared secret is the $j$-invariant of $E_{AB}$

# Oblivious Pseudorandom Functions from Isogenies [BKW20]

Client
Server

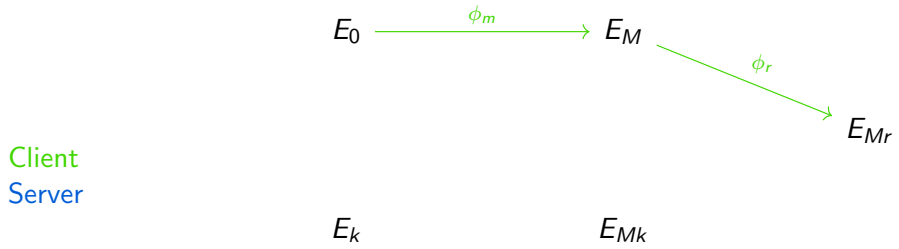


# Oblivious Pseudorandom Functions from Isogenies [BKW20]

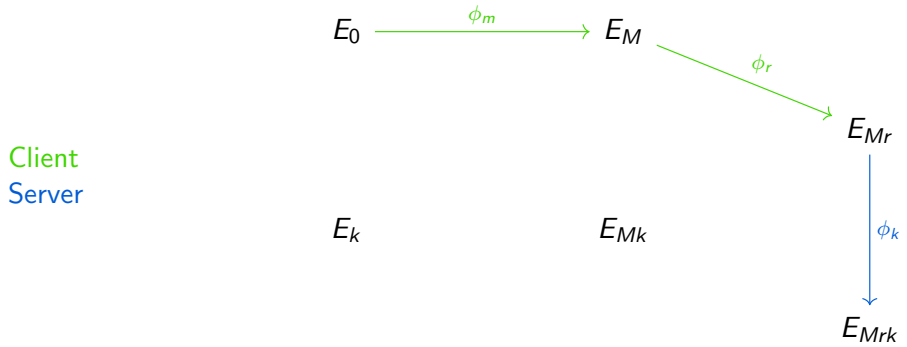

Client
Server

$E_k$ $E_{Mk}$

# Oblivious Pseudorandom Functions from Isogenies [BKW20]

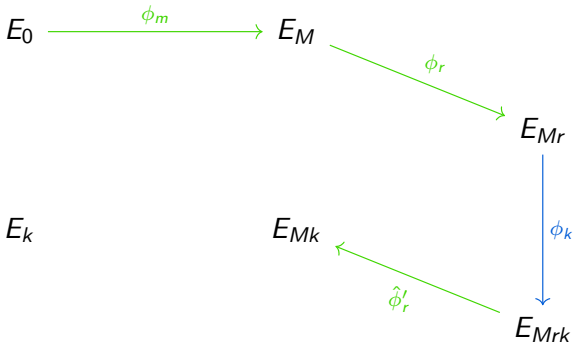


# Oblivious Pseudorandom Functions from Isogenies [BKW20]



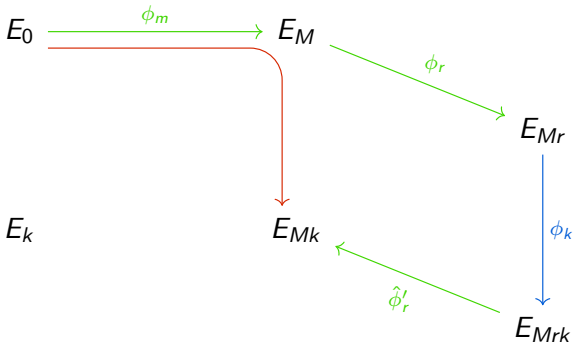
# Oblivious Pseudorandom Functions from Isogenies [BKW20]

Client
Server



# Oblivious Pseudorandom Functions from Isogenies [BKW20]

Client
Server



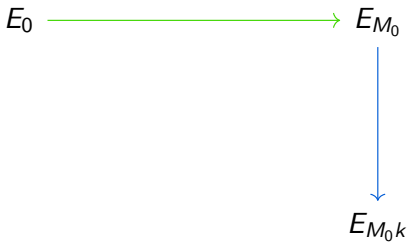

$$f(k, m) = H(m, j(E_{Mk}), \mathrm{pk})$$

# Pseudorandomness of an Oblivious PRF

- An attacker should not be able to evaluate the OPRF without the server's help even after multiple queries

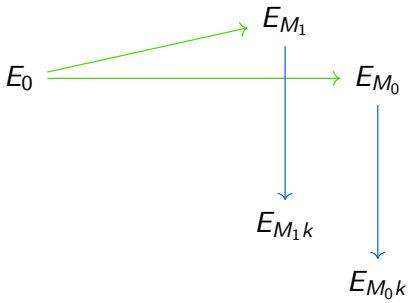
# Pseudorandomness of an Oblivious PRF

- An attacker should not be able to evaluate the OPRF without the server's help even after multiple queries



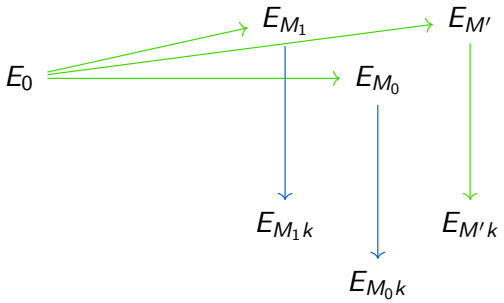
# Pseudorandomness of an Oblivious PRF

- An attacker should not be able to evaluate the OPRF without the server's help even after multiple queries



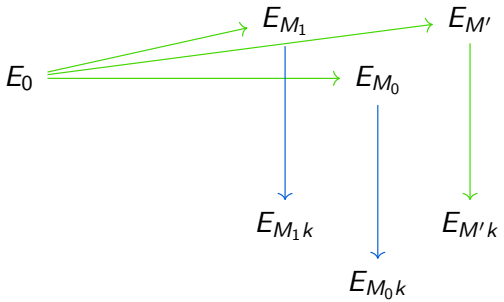
# Pseudorandomness of an Oblivious PRF

- An attacker should not be able to evaluate the OPRF without the server's help even after multiple queries

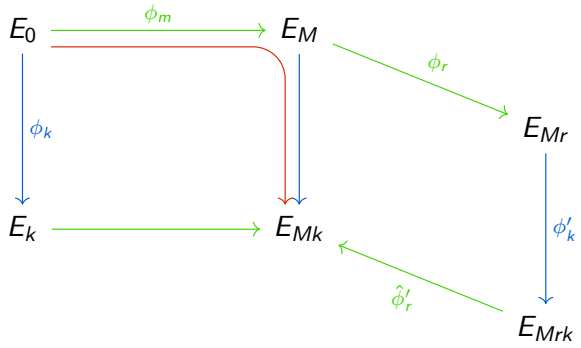


# Pseudorandomness of an Oblivious PRF

- An attacker should not be able to evaluate the OPRF without the server's help even after multiple queries
- Pseudorandomness of [BKW20] is based on a new 'auxiliary one-more' assumption

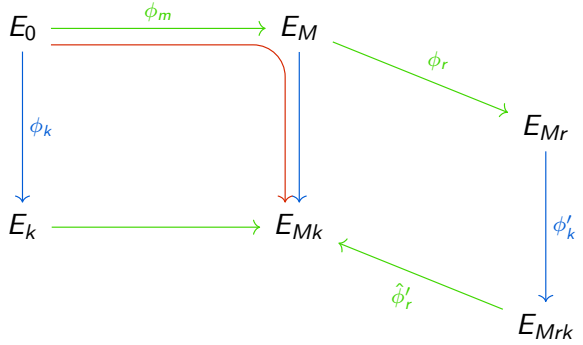


# Attacking the 'one-more' Assumption



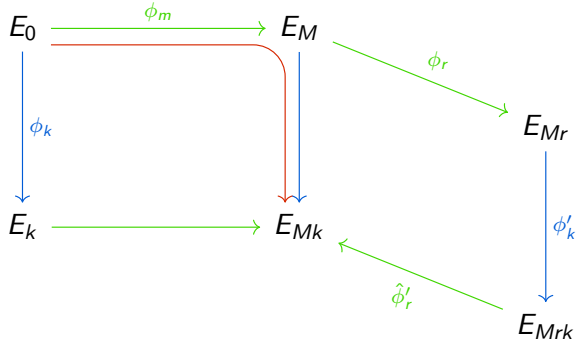

- Find $E_k$ and $\langle \phi_k(M) \rangle$ for some point $M \in E_0[2^n]$

# Attacking the 'one-more' Assumption




- Find $E_k$ and $\langle \phi_k(M) \rangle$ for some point $M \in E_0[2^n]$
- Combine multiple points to obtain $\phi_k(E_0[2^n])$ up to scalar multiplication

# Attacking the 'one-more' Assumption




- Find $E_k$ and $\langle \phi_k(M) \rangle$ for some point $M \in E_0[2^n]$
- Combine multiple points to obtain $\phi_k(E_0[2^n])$ up to scalar multiplication
- Given point $P \in E_0[2^n]$, compute $\langle \phi_k(P) \rangle$ and finally $E_k / \langle \phi_k(P) \rangle = E_{Pk}$

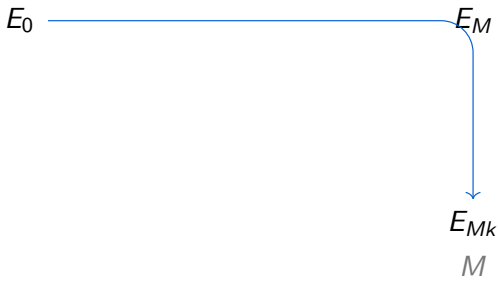
# A Polytime Attack

## Recovering points on $E_k$

$E_0$

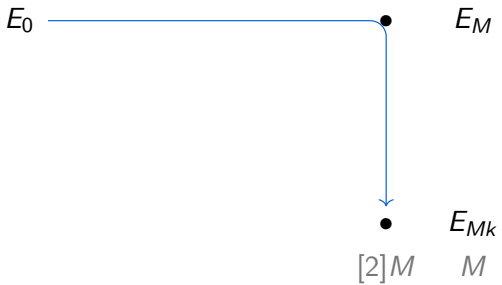
# A Polytime Attack

## Recovering points on $E_k$



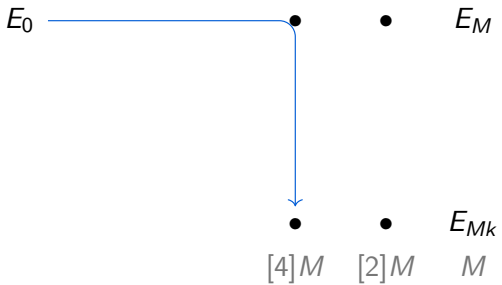
# A Polytime Attack

## Recovering points on $E_k$



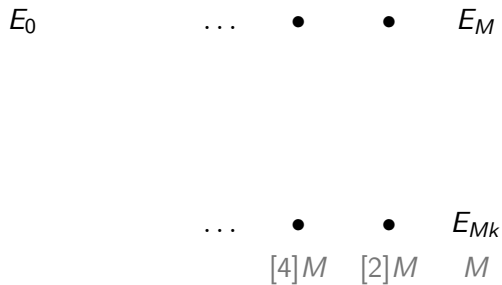
# A Polytime Attack

## Recovering points on $E_k$



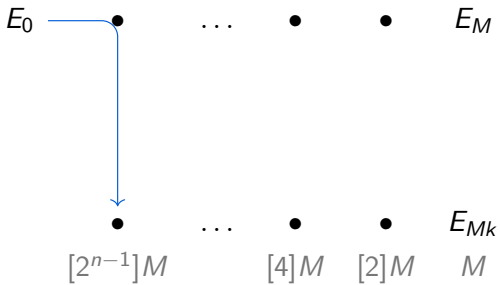
# A Polytime Attack

## Recovering points on $E_k$



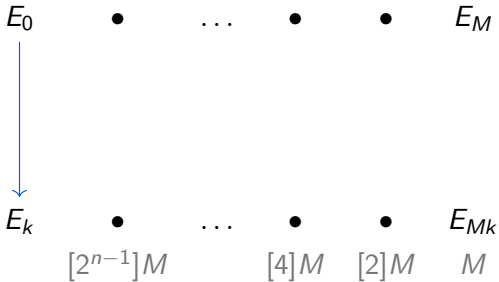
# A Polytime Attack

## Recovering points on $E_k$



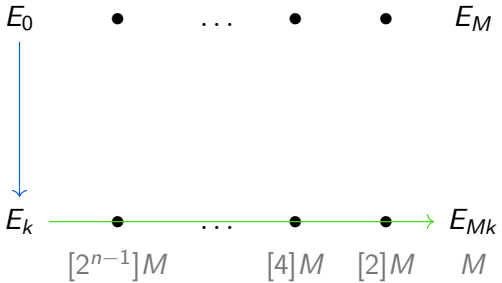
# A Polytime Attack

## Recovering points on $E_k$



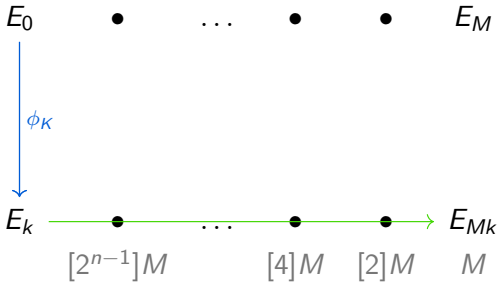
# A Polytime Attack

## Recovering points on $E_k$



# A Polytime Attack

## Recovering points on $E_k$




$$\ker \phi = \langle \phi_K(M) \rangle$$

# A Polytime Attack

## Combining the points

Given $M$ on $E_0[2^n]$, we can recover $\langle \phi_K(M) \rangle$

# A Polytime Attack

## Combining the points

Given $M$ on $E_0[2^n]$, we can recover $\langle \phi_K(M) \rangle \quad \Rightarrow \quad$ we can recover $[\alpha]\phi_K(M)$

# A Polytime Attack

## Combining the points

Given $M$ on $E_0[2^n]$, we can recover $\langle \phi_K(M) \rangle \quad \Rightarrow \quad$ we can recover $[\alpha]\phi_K(M)$

We query on $M, N, M+N$ and obtain

$$M' = [\alpha]\phi_K(M)$$
$$N' = [\beta]\phi_K(N)$$
$$R' = [\gamma]\phi_K(M+N) = [a]M' + [b]N'$$

# A Polytime Attack

## Combining the points

Given $M$ on $E_0[2^n]$, we can recover $\langle \phi_K(M) \rangle \quad \Rightarrow \quad$ we can recover $[\alpha]\phi_K(M)$

We query on $M, N, M+N$ and obtain

$$\left.\begin{aligned} M' &= [\alpha]\phi_K(M) \\ N' &= [\beta]\phi_K(N) \\ R' &= [\gamma]\phi_K(M+N) = [a]M' + [b]N' \end{aligned}\right\} \Rightarrow \frac{\alpha}{\beta} = \frac{b}{a}$$

# A Polytime Attack

## Combining the points

Given $M$ on $E_0[2^n]$, we can recover $\langle \phi_K(M) \rangle \quad \Rightarrow \quad$ we can recover $[\alpha]\phi_K(M)$

We query on $M, N, M+N$ and obtain

$$\left.\begin{aligned} M' &= [\alpha]\phi_K(M) \\ N' &= [\beta]\phi_K(N) \\ R' &= [\gamma]\phi_K(M+N) = [a]M' + [b]N' \end{aligned}\right\} \Rightarrow \frac{\alpha}{\beta} = \frac{b}{a}$$

**Breaking the assumption**

Given any $P = [x]M + [y]N$, we can compute $\langle \phi_K(P) \rangle = \langle [x]M' + [y]\frac{\alpha}{\beta}N' \rangle$

# A Polytime Attack

## Results

- $O(\lambda)$ queries recover $\langle \phi_K(M) \rangle$ for any $M$ in $E_0[2^n]$
- With three subgroups, we can compute $\langle \phi_K(P) \rangle$ for any $P$ without further interactions
- This breaks the 'one-more' assumption

# A Polytime Attack

## Results

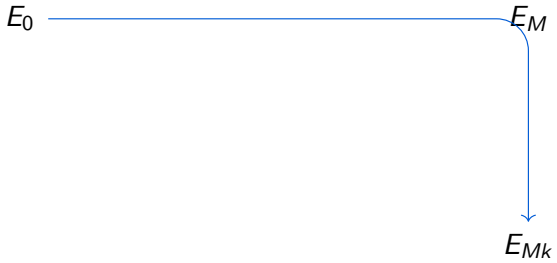
- $O(\lambda)$ queries recover $\langle \phi_K(M) \rangle$ for any $M$ in $E_0[2^n]$
- With three subgroups, we can compute $\langle \phi_K(P) \rangle$ for any $P$ without further interactions
- This breaks the 'one-more' assumption

But

- It is easy to check that query points have full order

# A Subexponential Attack

## Using full-order queries



# A Subexponential Attack

## Using full-order queries

$E_0$

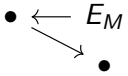
$\bullet \longleftarrow E_M$

$E_{Mk}$

# A Subexponential Attack

## Using full-order queries

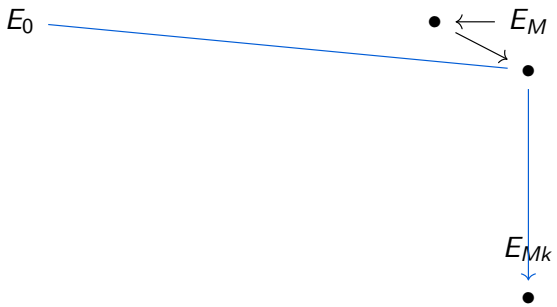
$E_0$




$E_{Mk}$

# A Subexponential Attack

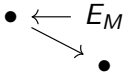
## Using full-order queries

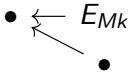


# A Subexponential Attack

## Using full-order queries

$E_0$



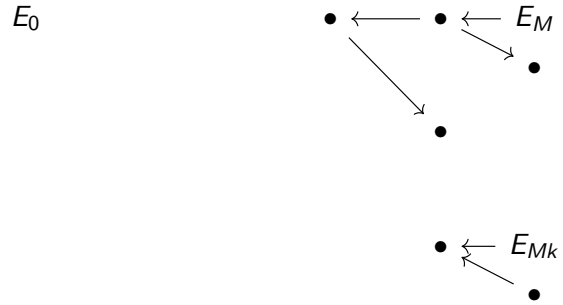



# A Subexponential Attack

## Using full-order queries

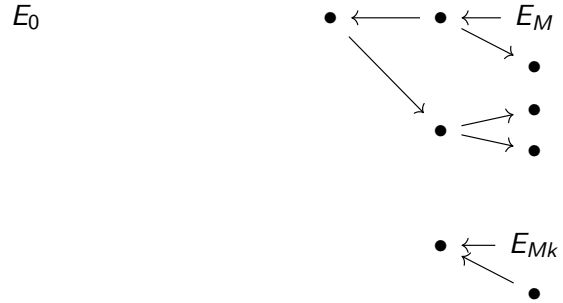
# A Subexponential Attack

## Using full-order queries



# A Subexponential Attack

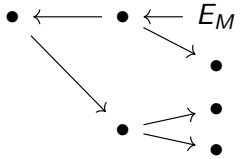
## Using full-order queries

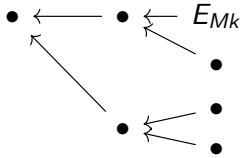


# A Subexponential Attack

## Using full-order queries

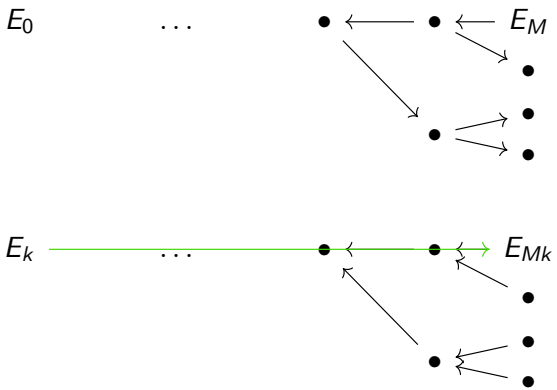
$E_0$



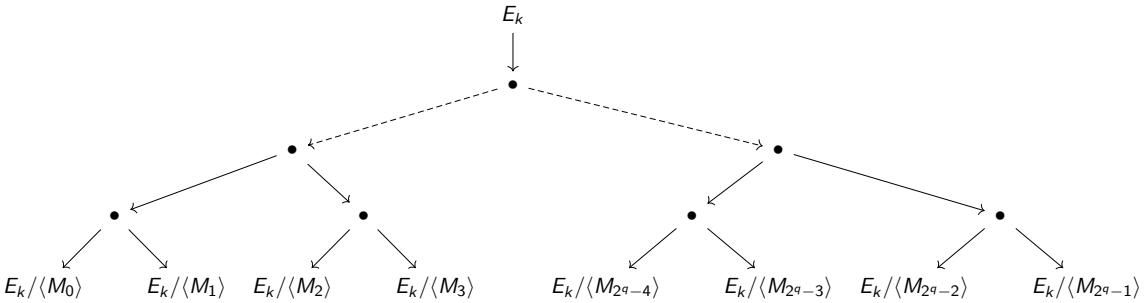
# A Subexponential Attack

## Using full-order queries



# A Subexponential Attack

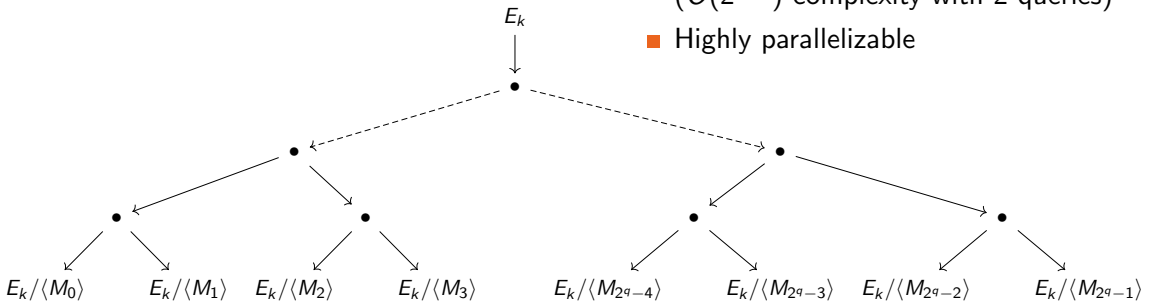
## Building a tree



# A Subexponential Attack

## Building a tree

- Queries/complexity trade-offs ($O(2^{\lambda/3})$ complexity with 2 queries)
- Highly parallelizable



# A Subexponential Attack

The full attack:

- Use the binary tree to recover points on $E_k$
- Second part of the attack same as polytime attack
- Subexponential complexity for balanced trade-offs

# A Subexponential Attack

The full attack:

- Use the binary tree to recover points on $E_k$
- Second part of the attack same as polytime attack
- Subexponential complexity for balanced trade-offs

Countermeasures:

- No obvious countermeasures
- Increase the parameter size? $\Rightarrow$ very large degrees
- New efficient solutions?

# Implementation Results

| Parameters | | | | MITM | | Running Time |
|---|---|---|---|---|---|---|
| $\log p$ | $\lambda$ | $n$ | $q$ | Distance | Memory (kB) | (s) |
| 112 | 8 | 20 | 3 | 8 | 3.5 | 15 |
| 216 | 16 | 40 | 6 | 10 | 13.8 | 212 (3.53 m) |
| 413 | 32 | 80 | 8 | 16 | 211.4 | 1,371 (22.85 m) |
| 859 | 67 | 169 | 11 | 26 | 14,073 | 163,869 (1.89 d) |
| 1,614 | 128 | 320 | 18 | 40 | 3,384,803 | *174,709,440 (5.54 y)* |

Available at `https://github.com/isogenists/isogeny-OPRF`

# The Starting Curve

Who chooses $E_0$?

- The client
- A third-party
- The server
- Known curve ($j(E_0) = 1728$)
- Trusted setup

# The Starting Curve

Who chooses $E_0$?

- The client
- A third-party

} can backdoor $E_0$ $\Rightarrow$ key-recovery attack on the server

- The server
- Known curve ($j(E_0) = 1728$)
- Trusted setup

# The Starting Curve

Who chooses $E_0$?

- The client
- A third-party

} can backdoor $E_0$ $\Rightarrow$ key-recovery attack on the server

- The server
- Known curve ($j(E_0) = 1728$)
- Trusted setup

} breaks the *Supersingular Isogeny Collision* assumption

# The Starting Curve

Who chooses $E_0$?

- The client
- A third-party

} can backdoor $E_0$ $\Rightarrow$ key-recovery attack on the server

- The server
- Known curve ($j(E_0) = 1728$)
- **Trusted setup**

} breaks the *Supersingular Isogeny Collision* assumption

# Conclusion

- Two attacks on 'one-more' assumption and the pseudorandomness of Boneh et al.'s OPRF
- A proof of concept implementation of the attack
- Need for a trusted setup
- CSIDH-based OPRF construction is not affected by the attack

Paper available at `https://ia.cr/2021/706`

# References

[BKW20] Dan Boneh, Dmitry Kogan, and Katharine Woo. Oblivious pseudorandom functions from isogenies. In *Advances in Cryptology - ASIACRYPT 2020 - 26th International Conference on the Theory and Application of Cryptology and Information Security, Daejeon, South Korea, December 7-11, 2020, Proceedings, Part II*, pages 520–550, 2020.