Cryptanalysis of an Oblivious PRF from Supersingular Isogenies

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

July 2022

IBM Isogeny Day

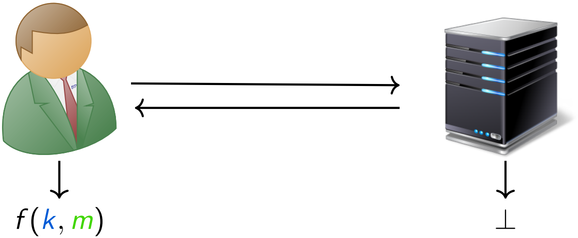
Content

- Definition of (V)OPRFs
- Applications
  - OPAQUE
- Classical construction
- OPRFs from group actions
- SIDH-based OPRF
- Cryptanalytic results
  - Polytime and subexponential attacks on SIDH-based version
  - Requirement for trusted setup

Oblivious Pseudorandom Function (OPRF)

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

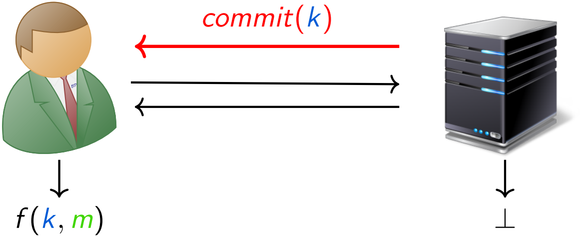
- The client learns $f(k, m)$, one evaluation of a PRF on a chosen input
- The server learns nothing about m



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



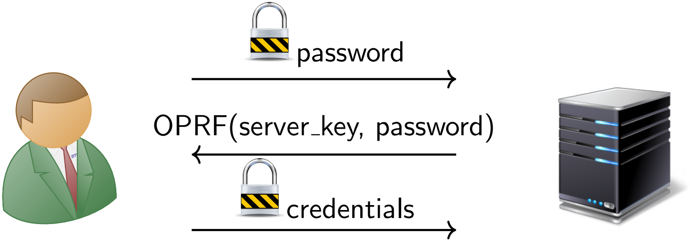
- An OPRF is called *verifable*, if the server proves to the client that output was computed using the key k

OPAQUE: OPRF + PAKE

- Use passwords that never leave your device

How to check a password that you have never seen?

Registration Phase:

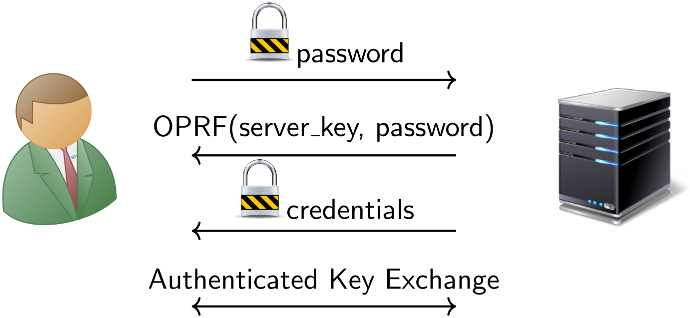


OPAQUE: OPRF + PAKE

- Use passwords that never leave your device

How to check a password that you have never seen?

Login Phase:



Classical Construction

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 $H_2(m, v)$		

Classical Construction

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]

Naor-Reingold OPRF from group actions [BKW20]

Definition

Let S be a set, $s_0 \in S$ and G be a finite abelian group acting on S free and transitively. The Naor-Reingold PRF with key space $\mathcal{K} = G^{n+1}$ and input space $\mathcal{M} = \{0,1\}^n$ is

$$F_{\mathrm{NR}}\big((k_0, k_1, \ldots, k_n), (m_1, \ldots, m_n)\big) = (k_0 k_1^{m_1} \cdots k_n^{m_n}) \cdot s_0$$

- security of PRF relies on group-action DDH assumption

Naor-Reingold OPRF from group actions [BKW20]

Definition

Let $Ell_p(\mathcal{O})$ be the set of supersingular curves over $\mathbb{F}_p$ with endomorphism ring $\mathcal{O}$, $E_0 \in Ell_p(\mathcal{O})$ and $Cl(\mathcal{O})$ the class group acting freely and transitively on $Ell_p(\mathcal{O})$.
The Naor-Reingold PRF with key space $\mathcal{K} = Cl(\mathcal{O})^{n+1}$ and input space $\mathcal{M} = \{0,1\}^n$ is

$$F_{\mathrm{NR}}\big(([\mathfrak{a}_0],[\mathfrak{a}_1],\ldots,[\mathfrak{a}_n]),(m_1,\ldots,m_n)\big) = ([\mathfrak{a}_0][\mathfrak{a}_1]^{m_1}\cdots[\mathfrak{a}_n]^{m_n})\cdot E_0$$

Naor-Reingold OPRF from group actions [BKW20] contd.

$$F_{\mathrm{NR}}\big((k_0, k_1, \ldots, k_n), (m_1, \ldots, m_n)\big) = \left(k_0 k_1^{m_1} \cdots k_n^{m_n}\right) \cdot s_0$$

Naor-Reingold OPRF from group actions [BKW20] contd.

$$F_{\mathrm{NR}}\big((k_0, k_1, \dots, k_n), (m_1, \dots, m_n)\big) = \left(k_0 k_1^{m_1} \cdots k_n^{m_n}\right) \cdot s_0$$

Client		Server
$(m_1, \dots, m_n \in \{0,1\}^n)$		$((k_0, k_1, \dots, k_n) \in G^{n+1})$
		$r_i \leftarrow_R G, i = 1, \dots, n$
	OT :	
	$\xleftarrow{r_i,\text{ if } m_i=0}$	
	$\xleftarrow{k_i r_i,\text{ if } m_i=1}$	
Store output as b_i		$s' \leftarrow \left(k_0 \prod_i r_i^{-1}\right) \cdot s_0$
	$\xleftarrow{s'}$	
Compute $\left(\prod_i b_i\right) \cdot s'$ $= \left(k_0 k_1^{m_1} \cdots k_n^{m_n}\right) \cdot s_0$		

SIDH-based OPRF [BKW20]

Client
Server

$$\begin{tikzcd}
E_0 \arrow[r, "\phi_m"] \arrow[d, "\phi_k"] & E_M \arrow[d] \\
E_k \arrow[r] & E_{Mk}
\end{tikzcd}$$

SIDH-based OPRF [BKW20]

$$E_0 \xrightarrow{\phi_m} E_M$$

Client
Server

$$E_{Mk}$$

SIDH-based OPRF [BKW20]

$$\begin{tikzcd}
E_0 \arrow[r, "\phi_m"] & E_M \arrow[dr, "\phi_r"] & \\
 & & E_{Mr} \\
 & E_{Mk} &
\end{tikzcd}$$

Client
Server

SIDH-based OPRF [BKW20]

Client
Server

$$
\begin{tikzcd}
E_0 \arrow[r, "\phi_m"] & E_M \arrow[dr, "\phi_r"] & \\
 & & E_{Mr} \arrow[dd, "\phi'_k"] \\
 & E_{Mk} & \\
 & & E_{Mrk}
\end{tikzcd}
$$

SIDH-based OPRF [BKW20]

$$
\begin{tikzcd}
E_0 \arrow[r, "\phi_m"] & E_M \arrow[dr, "\phi_r"] & \\
 & & E_{Mr} \arrow[d, "\phi'_k"] \\
 & E_{Mk} & E_{Mrk} \arrow[l, "\hat{\phi}'_r"]
\end{tikzcd}
$$

Client
Server

$$\begin{tikzcd}
E_0 \arrow[r, "\phi_m"] \arrow[dr, bend right] & E_M \arrow[dr, "\phi_r"] & \\
 & E_{Mk} & E_{Mr} \arrow[d, "\phi'_k"] \\
 & & E_{Mrk} \arrow[ul, "\hat{\phi}'_r"]
\end{tikzcd}$$

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

$$\begin{tikzcd} E_0 \arrow[r] & E_{M_0} \arrow[d] \\ & E_{M_0 k} \end{tikzcd}$$

Pseudorandomness of an Oblivious PRF

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

$$
\begin{tikzcd}
 & E_{M_1} \arrow[dd] & \\
E_0 \arrow[ur] \arrow[rr] & & E_{M_0} \arrow[dd] \\
 & E_{M_1 k} & \\
 & & E_{M_0 k}
\end{tikzcd}
$$

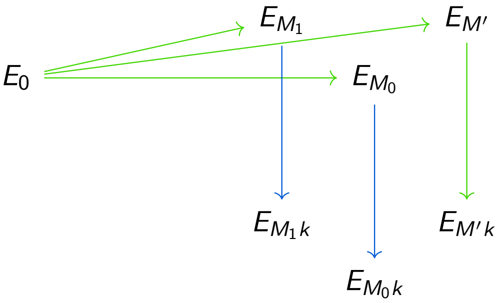
Pseudorandomness of an Oblivious PRF

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

$$\begin{tikzcd}
& & E_{M_1} \arrow[dd] & & E_{M'} \arrow[dd] \\
E_0 \arrow[urr] \arrow[urrrr] \arrow[rrr] & & & E_{M_0} \arrow[dd] & \\
& & E_{M_1 k} & & E_{M' k} \\
& & & E_{M_0 k} &
\end{tikzcd}$$

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



Breaking the Pseudorandomness

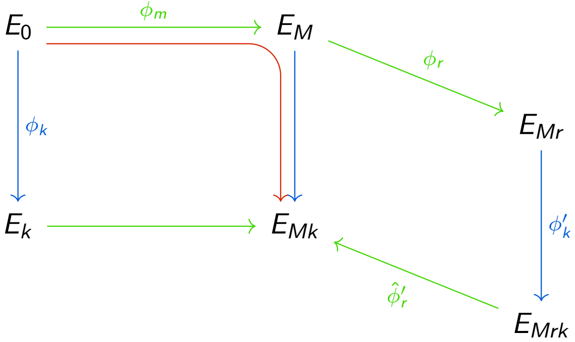
$$
\begin{tikzcd}
E_0 \arrow[r, "\phi_m"] \arrow[dd, "\phi_k"] \arrow[rdd, bend left] & E_M \arrow[dd] \arrow[dr, "\phi_r"] & \\
 & & E_{Mr} \arrow[dd, "\phi'_k"] \\
E_k \arrow[r] & E_{Mk} & \\
 & & E_{Mrk} \arrow[ul, "\hat{\phi}'_r"]
\end{tikzcd}
$$

Breaking the Pseudorandomness

$$
\begin{tikzcd}
E_0 \arrow[r, "\phi_m"] \arrow[dd, "\phi_k"] \arrow[ddr, bend left] & E_M \arrow[dr, "\phi_r"] \arrow[dd] & \\
 & & E_{Mr} \arrow[dd, "\phi'_k"] \\
E_k \arrow[r] & E_{Mk} & \\
 & & E_{Mrk} \arrow[ul, "\hat{\phi}'_r"]
\end{tikzcd}
$$

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

Breaking the Pseudorandomness



- 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

$$\begin{tikzcd}
E_0 \arrow[r, "\phi_m"] \arrow[d, "\phi_k"] \arrow[dr, bend left] & E_M \arrow[d] \arrow[r, "\phi_r"] & E_{Mr} \arrow[d, "\phi'_k"] \\
E_k \arrow[r] & E_{Mk} & E_{Mrk} \arrow[l, "\hat{\phi}'_r"]
\end{tikzcd}$$

- 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 thus $E_k / \langle \phi_k(P) \rangle = E_{Pk}$

A Polytime Attack

Recovering subgroups $\langle \phi_k(M) \rangle \subset E_k$

E_0

A Polytime Attack

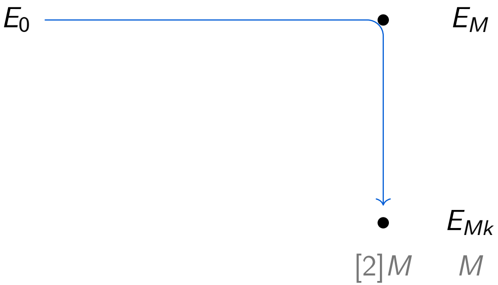
Recovering subgroups $\langle \phi_k(M) \rangle \subset E_k$

$$
\begin{tikzcd}
E_0 \arrow[rr] & & E_M \arrow[d] \\
& & E_{Mk}
\end{tikzcd}
$$

M

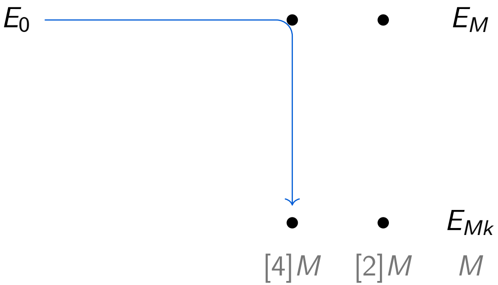
A Polytime Attack

Recovering subgroups $\langle \phi_k(M) \rangle \subset E_k$



A Polytime Attack

Recovering subgroups $\langle \phi_k(M) \rangle \subset E_k$



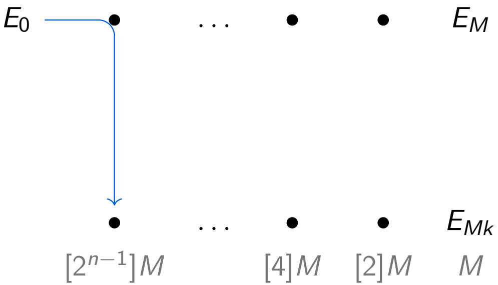
A Polytime Attack

Recovering subgroups $\langle \phi_k(M) \rangle \subset E_k$

$$
\begin{array}{ccccc}
E_0 & \ldots & \bullet & \bullet & E_M \\
 & & & & \\
 & \ldots & \bullet & \bullet & E_{Mk} \\
 & & [4]M & [2]M & M
\end{array}
$$

A Polytime Attack

Recovering subgroups $\langle \phi_k(M) \rangle \subset E_k$



A Polytime Attack

Recovering subgroups $\langle \phi_k(M) \rangle \subset E_k$

$$
\begin{tikzcd}
E_0 \arrow[d] & \bullet & \ldots & \bullet & \bullet & E_M \\
E_k & \bullet & \ldots & \bullet & \bullet & E_{Mk} \\
 & {[2^{n-1}]M} & & {[4]M} & {[2]M} & M
\end{tikzcd}
$$

A Polytime Attack

Recovering subgroups $\langle \phi_k(M) \rangle \subset E_k$

$$
\begin{tikzcd}
E_0 \arrow[d] & \bullet & \cdots & \bullet & \bullet & E_M \\
E_k \arrow[rrrrr] & \bullet & \cdots & \bullet & \bullet & E_{Mk} \\
& {[2^{n-1}]M} & & {[4]M} & {[2]M} & M
\end{tikzcd}
$$

A Polytime Attack

Recovering subgroups $\langle \phi_k(M) \rangle \subset E_k$

$$\begin{tikzcd} E_0 \arrow[d, "\phi_k"] & \bullet & \cdots & \bullet & \bullet & E_M \\ E_k \arrow[rrrrr] & \bullet & \cdots & \bullet & \bullet & E_{Mk} \\ & {[2^{n-1}]M} & & {[4]M} & {[2]M} & M \end{tikzcd}$$

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

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 $\langle \phi_k(M) \rangle$

$\Rightarrow$ can recover $[\alpha]\phi_k(M)$ for odd α

A Polytime Attack

Combining the points

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

$\Rightarrow$ can recover $[\alpha]\phi_k(M)$ for odd α

For basis $E_0[2^n] = \langle M, N \rangle$, we recover

$$\begin{aligned} M' &:= [\alpha]\phi_k(M) \\ N' &:= [\beta]\phi_k(N) \\ R' &:= [\gamma]\phi_k(M+N) \end{aligned}$$

A Polytime Attack

Combining the points

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

$\Rightarrow$ can recover $[\alpha]\phi_k(M)$ for odd α

For basis $E_0[2^n] = \langle M, N \rangle$, we recover

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

A Polytime Attack

Combining the points

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

$\Rightarrow$ can recover $[\alpha]\phi_k(M)$ for odd α

For basis $E_0[2^n] = \langle M, N \rangle$, we recover

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

Breaking Pseudorandomness

Given any $P = [x]M + [y]N \in E_0[2^n]$, we can compute

$$\langle \phi_k(P) \rangle = \langle [x]M' + [y]\left[\frac{\alpha}{\beta}\right] N' \rangle$$

A Polytime Attack

Results

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

$$
\begin{tikzcd}
E_0 \arrow[r, "\phi_m"] \arrow[dd, "\phi_k"] \arrow[ddr, bend left] & E_M \arrow[dr, "\phi_r"] \arrow[dd] & \\
 & & E_{Mr} \arrow[dd, "\phi'_k"] \\
E_k \arrow[r] & E_{Mk} & \\
 & & E_{Mrk} \arrow[ul, "\hat{\phi}'_r"]
\end{tikzcd}
$$

A Polytime Attack
Results

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

But

- It can be checked that query points have full order

A Subexponential Attack

Using full-order queries

$$\begin{tikzcd} E_0 \arrow[r, -] & E_M \arrow[d] \\ & E_{Mk} \end{tikzcd}$$

A Subexponential Attack

Using full-order queries

E_0 $\qquad\qquad\qquad\qquad\qquad \bullet \longleftarrow E_M$

E_{Mk}

A Subexponential Attack

Using full-order queries

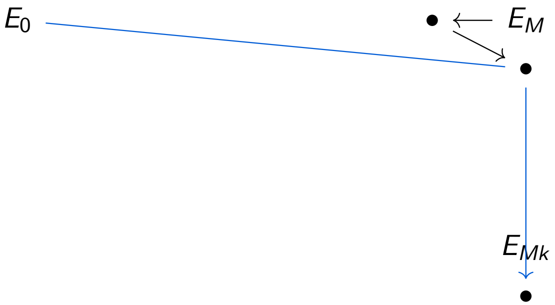
E_0

E_M

E_{Mk}

A Subexponential Attack

Using full-order queries



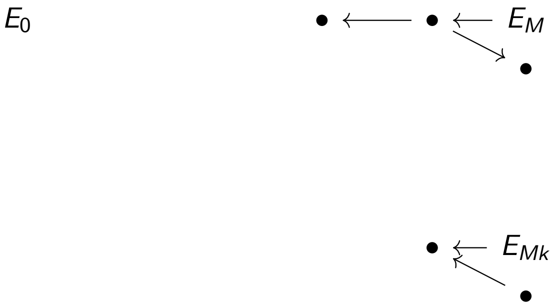
A Subexponential Attack

Using full-order queries

$$\begin{tikzcd}
E_0 & & & \bullet \arrow[dr] & E_M \arrow[l] \\
& & & & \bullet \\
& & & & \\
& & & \bullet & E_{Mk} \arrow[l] \\
& & & & \bullet \arrow[ul]
\end{tikzcd}$$

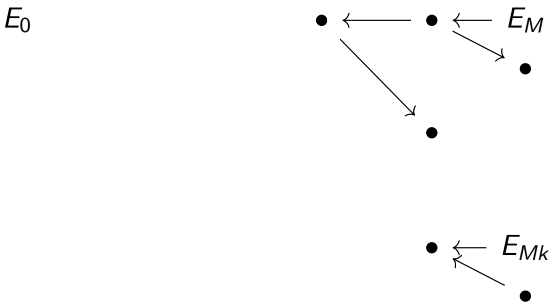
A Subexponential Attack

Using full-order queries



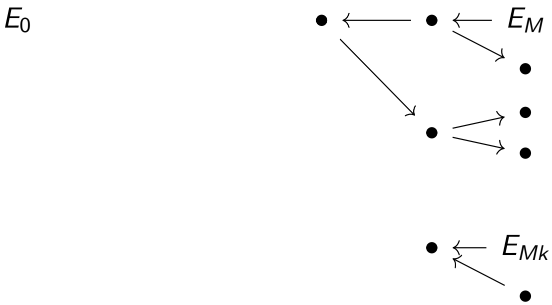
A Subexponential Attack

Using full-order queries



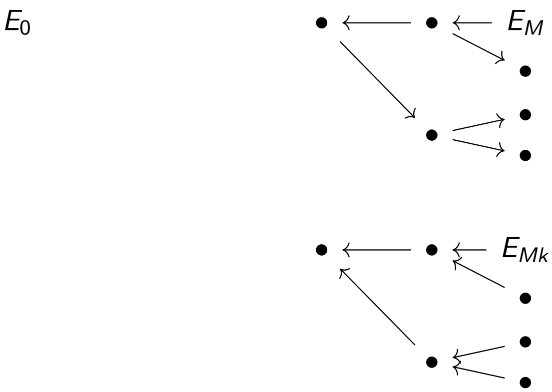
A Subexponential Attack

Using full-order queries



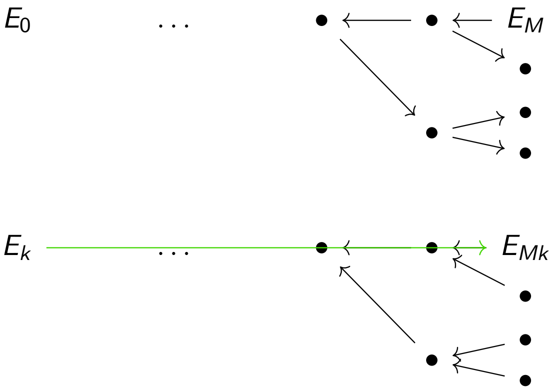
A Subexponential Attack

Using full-order queries



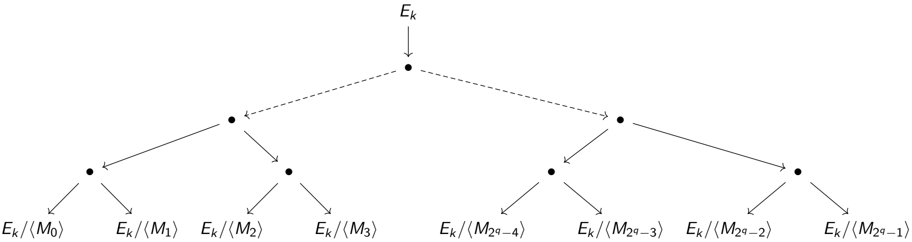
A Subexponential Attack

Using full-order queries



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 subgroup generating $E_k \to E_{Mk}$
- 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 subgroup generating $E_k \to E_{Mk}$
- 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		Time
$\log p$	λ	n	q	Distance	Memory (kB)	
112	8	20	3	8	3.5	15s
216	16	40	6	10	13.8	3.53 m
413	32	80	8	16	211.4	22.85 m
859	67	169	11	26	14,073	1.89 d
1,614	128	320	18	40	3,384,803	5.54 y

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

The Starting Curve

$$
\begin{tikzcd}
E_0 \arrow[r, "\phi_m"] \arrow[dd, "\phi_k"] \arrow[ddr, bend left] & E_M \arrow[dd] \arrow[dr, "\phi_r"] & \\
 & & E_{Mr} \arrow[dd, "\phi'_k"] \\
E_k \arrow[r] & E_{Mk} & \\
 & & E_{Mrk} \arrow[ul, "\hat{\phi}'_r"]
\end{tikzcd}
$$

Who chooses E_0?

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

The Starting Curve

$$
\begin{tikzcd}
E_0 \arrow[r, "\phi_m"] \arrow[dd, "\phi_k"] \arrow[ddr, bend left] & E_M \arrow[dr, "\phi_r"] \arrow[dd] & \\
& & E_{Mr} \arrow[dd, "\phi'_k"] \\
E_k \arrow[r] & E_{Mk} & \\
& & E_{Mrk} \arrow[uul, "\hat{\phi}'_r"]
\end{tikzcd}
$$

Who chooses E_0?

- The client
- A third-party

$\Big\}$ can backdoor E_0 allowing to recover k

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

The Starting Curve

$$\begin{tikzcd}
E_0 \arrow[r, "\phi_m"] \arrow[dd, "\phi_k"] \arrow[ddr, bend left] & E_M \arrow[dd] \arrow[dr, "\phi_r"] & \\
& & E_{Mr} \arrow[dd, "\phi'_k"] \\
E_k \arrow[r] & E_{Mk} & \\
& & E_{Mrk} \arrow[ul, "\hat{\phi}'_r"]
\end{tikzcd}$$

Who chooses E_0?

- The client
- A third-party

$\}$ can backdoor E_0 allowing to recover k

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

$\}$ breaks *Collision* assumption

- Trusted setup

The Starting Curve

$$\begin{tikzcd}
E_0 \arrow[r, "\phi_m"] \arrow[dd, "\phi_k"] \arrow[ddr] & E_M \arrow[dd] \arrow[dr, "\phi_r"] & \\
 & & E_{Mr} \arrow[dd, "\phi'_k"] \\
E_k \arrow[r] & E_{Mk} & \\
 & & E_{Mrk} \arrow[ul, "\hat{\phi}'_r"]
\end{tikzcd}$$

Who chooses E_0?

$$\left.\begin{array}{l}\text{The client}\\ \text{A third-party}\end{array}\right\}\ \text{can backdoor } E_0 \text{ allowing to recover } k$$

$$\left.\begin{array}{l}\text{The server}\\ \text{Known curve } (j(E_0) = 1728)\end{array}\right\}\ \text{breaks } \textit{Collision} \text{ assumption}$$

- **Trusted setup**

Conclusion

- Two attacks on pseudorandomness of SIDH-based OPRF by Boneh, Kogan and Woo
- A proof of concept implementation of the attack
- Need for a trusted setup
- Can we build better post-quantum OPRFs?

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