On Adaptive Attacks against Jao-Urbanik’s Isogeny-Based Protocol

https://ia.cr/2020/244

Andrea Basso\textsuperscript{1}, Péter Kutas\textsuperscript{1}, Simon-Philipp Merz\textsuperscript{2}, Christophe Petit\textsuperscript{1}, Charlotte Weitkämper\textsuperscript{1}

University of Birmingham, UK
Royal Holloway, University of London, UK
Introduction

Protocols

SIDH \rightarrow k\text{-SIDH} \rightarrow JU scheme

FO \rightarrow SIKE

GPST \rightarrow DGLTZ \rightarrow [This work]

Attacks

$k$ instances

attacks

generalizes to

attacks

attacks

automorphisms
Our results

- We exploit the additional structure between curves in the JU scheme to achieve a nearly cubic speed-up when compared to the DGLTZ attack.
- Our attack does NOT break the JU scheme for the proposed parameters...
- ...but it shows that at the same security level the JU scheme requires almost twice the computations of $k$-SIDH to reduce the public-key size by 20%.
Additional Information
SIDH is a key-exchange protocol over supersingular elliptic curves defined over \( \mathbb{F}_{p^2} \), where \( p = 2^{e_A}3^{e_B}f \pm 1 \).

\[
\langle P_A, Q_A \rangle = E_0[2^{e_A}] \text{ and } \ker \phi_A = \langle P_A + [\alpha]Q_A \rangle,
\langle P_B, Q_B \rangle = E_0[3^{e_B}] \text{ and } \ker \phi_B = \langle P_B + [\beta]Q_B \rangle.
\]
GPST attack [3]

- Static secret keys in SIDH can be recovered by a dishonest participant Bob with the adaptive GPST attack
- An attacker uses the key exchange as an oracle to retrieve the static key $\alpha$ of Alice iteratively
- The oracle: returns true if $E_B / \langle R + [\alpha]S \rangle = E_{AB}$, where $R, S$ are the torsion points sent by the attacker Bob
- Sending malicious torsion points $R, S$ the dishonest participant Bob retrieves one bit of $\alpha$ per oracle query
- Countermeasure: Fujisaki-Okamoto or similar transform (as in SIKE)
\( k \)-SIDH [1]

\( k \)-SIDH avoids attacks such as GPST by performing \( k^2 \) instances of SIDH during a single execution of the static-static key exchange protocol.

Using each combination \( E_{A_i}, E_{B_j} \) for \( i, j = 1, \ldots, k \) of the two parties’ \( k \) different public curves yields shared secret

\[
\text{Hash}(j(E_{A_1B_1}), j(E_{A_1B_2}), \ldots, j(E_{A_kB_k})).
\]
The DGLTZ-attack on $k$-SIDH [2]

- The attacker queries with the same curve and same extra points for each SIDH instance
- New oracle: returns true if an attacker guesses all the common computed curves correctly
- First step: query with $(E_B, P, [1 + 2^{n-1}]Q)$, one has to query $6 \cdot 7^{k-1}$ times to get the first bit
- With this approach, even for $k = 2$, one needs an exponential number of queries
- DGLTZ solves the issue by computing the intermediate curves and additional points on those curves
- Computing these additional points requires $24^k$ queries
The Jao-Urbanik protocol [5]

The protocol improves on $k$-SIDH by using automorphisms to obtain three instances for each key.

- **Starting curve**: $E_0$, $j(E_0) = 0$, with non-trivial automorphism $\eta$ of order six
- **For any subgroup** $B \subset E_0$, $E_0/B \cong E_0/\eta(B) \cong E_0/\eta^2(B)$
- **Fix bases**:
  - $\{P_A, Q_A = \eta(P_A)\}$ of $E_0[2^{e_A}]$,
  - $\{P_B, Q_B = \eta(P_B)\}$ of $E_0[3^{e_B}]$
The Jao-Urbanik protocol II

- Alice and Bob perform SIDH-instance with public keys $(E_A, \phi_A(P_B), \phi_A(Q_B))$ and $(E_B, \phi_B(P_A), \phi_B(Q_A))$

- Alice and Bob obtain as shared secret information $(j$-invariants of)
  - $E_{A,B}$ as in standard SIDH
  - $E_{A,\eta(B)}$
  - $E_{A,\eta^2(B)}$ using $\eta$ during computation

E.g. Bob uses his secret key $\beta$ to compute

$$E_{A,\eta(B)} = E_A/\langle -\phi_B(P_A) + [\beta + 1]\phi_B(\eta(P_A)) \rangle$$ and

$$E_{A,\eta^2(B)} = E_A/\langle -[\beta + 1]\phi_B(P_A) + [\beta]\phi_B(\eta(P_A)) \rangle$$
Applying DGLTZ to Jao-Urbanik’s protocol

- DGLTZ treats each curve separately
- Secret kernel generators occurring in Jao-Urbanik protocol are not of the required form to straightforwardly apply DGLTZ
- If issues with kernel generators can be overcome, attacking the Jao-Urbanik protocol with \( k \) keys and \( 3k^2 \) SIDH-instances would require \( O(24^{3k}) \) queries

\[ \implies \text{This work uses relationships between curves and kernel generators to reduce number of queries.} \]
Our attack - First bit recovery

- Goal: get least significant bit $\alpha_0$ of Alice’s secret key $\alpha$, i.e. determine first curve on isogeny path $E_A \rightarrow E_0$.
- Query with $(E_B, [1 + 2^{n-1}]P_B, Q_B)$, so Alice computes all three 2-neighboring curves of $E/\langle 2A \rangle$.
- Underlying relationship between kernel generators of corresponding curves helps to match up triples of candidate curves instead of exhaustively searching over all possibilities.

\[
E_0 \quad \ldots \quad E_A' \cong E_{A,2} \quad \left\{ \begin{array}{c}
E_A \\
E/\langle 2A \rangle \cong E_{A,1}
\end{array} \right.
\]

$n - 1$ partial isogenies of $\phi_A$
Our attack - Pullbacks

- Main idea: Let $A$ be a secret kernel, let $E_{A,i}$, $E'_{A,i}$, $E''_{A,i}$ be the $i$th curves on the three corresponding paths. Then for all $i$, the curves $E_{A,i}$, $E'_{A,i}$, $E''_{A,i}$ are isomorphic.

- Instead of using the DGLTZ attack directly, we compute a pullback candidate for each curve and shift them with the corresponding isomorphisms.

- We query the oracle with these related points which saves a lot of time and exploits the extra structure of the scheme.
## Results

|                      | # SIDH instances | # keys per party | Attack cost          |
|----------------------|------------------|------------------|----------------------|
| **Jao-Urbanik**      |                  |                  |                      |
| with $k$ keys        | $3k^2$           | $k$              | $\mathcal{O}(\ell^{5k})$ |
|                      |                  |                  |                      |
| **$k$-SIDH**         |                  |                  |                      |
| with $\frac{5}{4}k$ keys | $1.56k^2$       | $\frac{5}{4}k$   | $\mathcal{O}(\ell^{5k})$ |

At the same security level, the JU scheme requires almost 2x computations to reduce the public key size by 20%.
References I

[1] Azarderakhsh, R., Jao, D., Leonardi, C.: Post-quantum static-static key agreement using multiple protocol instances. In: Adams, C., Camenisch, J. (eds.) Selected Areas in Cryptography – SAC 2017, vol. 10719, pp. 45–63. Springer International Publishing (2017), http://link.springer.com/10.1007/978-3-319-72565-9_3

[2] Dobson, S., Galbraith, S.D., LeGrow, J., Ti, Y.B., Zobernig, L.: An adaptive attack on 2-SIDH (2019), http://eprint.iacr.org/2019/890

[3] Galbraith, S.D., Petit, C., Shani, B., Ti, Y.B.: On the security of supersingular isogeny cryptosystems. In: Cheon, J.H., Takagi, T. (eds.) Advances in Cryptology – ASIACRYPT 2016. pp. 63–91. Lecture Notes in Computer Science, Springer (2016)
References II

[4] Jao, D., De Feo, L.: Towards quantum-resistant cryptosystems from supersingular elliptic curve isogenies. In: International Workshop on Post-Quantum Cryptography. pp. 19–34. Springer (2011)

[5] Urbanik, D., Jao, D.: New techniques for SIDH-based NIKE (accepted at MathCrypt 2018, to appear in J. Math. Cryptol.; personal communication)