# Post-Quantum Static-Static Key Agreement Using Multiple Protocol Instances

Reza Azarderakhsh [1] David Jao [2,3] Christopher Leonardi [2]

Department of Computer and Electrical Engineering and Computer Science, Florida Atlantic University

Department of Combinatorics and Optimization, University of Waterloo

evolutionQ, Inc., Waterloo, Ontario, Canada

August 2017

1. Isogeny-Based Key Agreement
   - Elliptic Curve Background
   - Jao-De Feo Key Agreement
   - Active Attack
2. Multiple Instances of Key Agreement
   - Protocol
   - Security
3. $k$-SIDH
   - Security
   - Conclusion

- An elliptic curve over a finite field $\mathbb{F}_{p^n}$,

$$E(\mathbb{F}_{p^n}) = \{(x, y) \in (\mathbb{F}_{p^n})^2 : y^2 = x^3 + ax + b\} \cup \{\mathcal{O}\},$$

  is a finite Abelian group.

- An elliptic curve over a finite field $\mathbb{F}_{p^n}$,

$$E(\mathbb{F}_{p^n}) = \{(x, y) \in (\mathbb{F}_{p^n})^2 : y^2 = x^3 + ax + b\} \cup \{\mathcal{O}\},$$

  is a finite Abelian group.

- The $m$-torsion subgroup $E[m] = \{P \in E(\overline{\mathbb{F}}_p) : [m]P = \mathcal{O}\}$.

- An elliptic curve over a finite field $\mathbb{F}_{p^n}$,

$$E(\mathbb{F}_{p^n}) = \{(x, y) \in (\mathbb{F}_{p^n})^2 : y^2 = x^3 + ax + b\} \cup \{\mathcal{O}\},$$

  is a finite Abelian group.

- The $m$-torsion subgroup $E[m] = \{P \in E(\overline{\mathbb{F}}_p) : [m]P = \mathcal{O}\}$.

- $E$ is called supersingular if $\forall r \in \mathbb{N}, E[p^r] = \{\mathcal{O}\}$ (otherwise $E$ is called ordinary).

- An elliptic curve over a finite field $\mathbb{F}_{p^n}$,

$$E(\mathbb{F}_{p^n}) = \{(x, y) \in (\mathbb{F}_{p^n})^2 : y^2 = x^3 + ax + b\} \cup \{\mathcal{O}\},$$

  is a finite Abelian group.

- The $m$-torsion subgroup $E[m] = \{P \in E(\overline{\mathbb{F}}_p) : [m]P = \mathcal{O}\}$.

- $E$ is called supersingular if $\forall r \in \mathbb{N}, E[p^r] = \{\mathcal{O}\}$ (otherwise $E$ is called ordinary).

- The $j$-invariant is a unique element of $\mathbb{F}_{p^n}$ associated to each $\overline{\mathbb{F}}_{p^n}$-isomorphism family of elliptic curves.

$$j(E) = 1728\frac{4a^3}{4a^3 + 27b^2} \in \mathbb{F}_{p^n}$$

- An isogeny $\phi : E \to E'$ over $\mathbb{F}_q$ is a non-constant rational map defined over $\mathbb{F}_q$ such that $\phi(\mathcal{O}_E) = \mathcal{O}_{E'}$, and is a group homomorphism from $E(\mathbb{F}_{p^n})$ to $E'(\mathbb{F}_{p^n})$.

- An isogeny $\phi : E \to E'$ over $\mathbb{F}_q$ is a non-constant rational map defined over $\mathbb{F}_q$ such that $\phi(\mathcal{O}_E) = \mathcal{O}_{E'}$, and is a group homomorphism from $E(\mathbb{F}_{p^n})$ to $E'(\mathbb{F}_{p^n})$.

- For each subgroup $G$ of $E$, there is up to isomorphism a unique isogeny $\phi$ with domain $E$ and kernel $G$. We denote the codomain curve by $E/G$.

- An isogeny $\phi : E \to E'$ over $\mathbb{F}_q$ is a non-constant rational map defined over $\mathbb{F}_q$ such that $\phi(\mathcal{O}_E) = \mathcal{O}_{E'}$, and is a group homomorphism from $E(\mathbb{F}_{p^n})$ to $E'(\mathbb{F}_{p^n})$.

- For each subgroup $G$ of $E$, there is up to isomorphism a unique isogeny $\phi$ with domain $E$ and kernel $G$. We denote the codomain curve by $E/G$.

- The degree, $\deg(\phi)$, is its degree as a rational map which is equal to the size of its kernel (for our purposes).

Global Parameters:

- Let $p = 2^m 3^n f \pm 1$, where $f$ is a small prime, and $E$ be a supersingular elliptic curve over $\mathbb{F}_{p^2}$.

Global Parameters:

- Let $p = 2^m 3^n f \pm 1$, where $f$ is a small prime, and $E$ be a supersingular elliptic curve over $\mathbb{F}_{p^2}$.

- Points $P_A, Q_A$ which generate the subgroup

$$E[2^m] \cong \mathbb{Z}/2^m\mathbb{Z} \times \mathbb{Z}/2^m\mathbb{Z}$$

Global Parameters:

- Let $p = 2^m 3^n f \pm 1$, where $f$ is a small prime, and $E$ be a supersingular elliptic curve over $\mathbb{F}_{p^2}$.
- Points $P_A, Q_A$ which generate the subgroup

$$E[2^m] \cong \mathbb{Z}/2^m\mathbb{Z} \times \mathbb{Z}/2^m\mathbb{Z}$$

- Points $P_B, Q_B$ which generate the subgroup

$$E[3^n] \cong \mathbb{Z}/3^n\mathbb{Z} \times \mathbb{Z}/3^n\mathbb{Z}$$

Key Generation:

- Alice:

$$\alpha \leftarrow_R \mathbb{Z}/2^m\mathbb{Z},$$
$$\phi_A \colon E \to E_A = E/\langle P_A + [\alpha]Q_A \rangle,$$
$$(R, S) \leftarrow (\phi_A(P_B),\ \phi_A(Q_B)).$$

Key Generation:

- Alice:

$$\alpha \leftarrow_R \mathbb{Z}/2^m\mathbb{Z},$$
$$\phi_A \colon E \to E_A = E/\langle P_A + [\alpha]Q_A \rangle,$$
$$(R, S) \leftarrow (\phi_A(P_B),\ \phi_A(Q_B)).$$

- Bob:

$$\beta \leftarrow_R \mathbb{Z}/3^n\mathbb{Z},$$
$$\phi_B \colon E \to E_B = E/\langle P_B + [\beta]Q_B \rangle,$$
$$(U, V) \leftarrow (\phi_B(P_A),\ \phi_B(Q_A)).$$

- Alice can compute:

$$\begin{aligned} E_B/\langle U + [\alpha]V \rangle &= E_B/\langle \phi_B(P_A) + [\alpha]\phi_B(Q_A) \rangle \\ &= E_B/\langle \phi_B(P_A + [\alpha]Q_A) \rangle \\ &= E/\langle P_B + [\beta]Q_B, P_A + [\alpha]Q_A \rangle \end{aligned}$$

- Alice can compute:

$$\begin{aligned} E_B/\langle U + [\alpha]V\rangle &= E_B/\langle \phi_B(P_A) + [\alpha]\phi_B(Q_A)\rangle \\ &= E_B/\langle \phi_B(P_A + [\alpha]Q_A)\rangle \\ &= E/\langle P_B + [\beta]Q_B, P_A + [\alpha]Q_A\rangle \end{aligned}$$

- Similarly Bob can compute:

$$\begin{aligned} E_A/\langle R + [\beta]S\rangle &= E_A/\langle \phi_A(P_B) + [\beta]\phi_A(Q_B)\rangle \\ &= E_A/\langle \phi_A(P_B + [\beta]Q_B)\rangle \\ &= E/\langle P_A + [\alpha]Q_A, P_B + [\beta]Q_B\rangle \end{aligned}$$

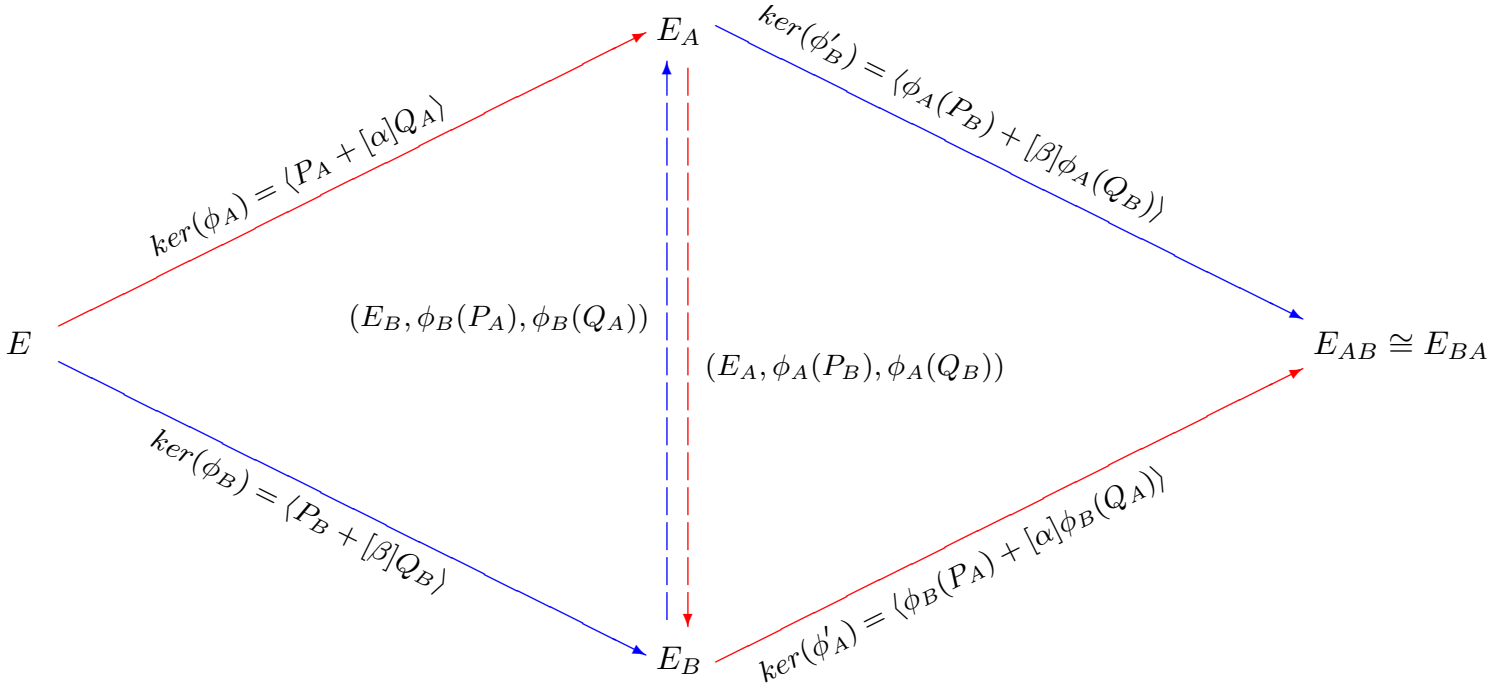


Figure: SIDH Key Agreement

- The shared secret is the $j$-invariant, $j(E_{AB}) \in \mathbb{F}_{p^2}$.

- This protocol is susceptible to an active attack when Alice reuses her key across multiple sessions.

- This protocol is susceptible to an active attack when Alice reuses her key across multiple sessions.

**Lemma**

[*Galbraith*, *Petit*, *Shani*, *Ti*, 2016] *Let* $P, Q \in E[2^m]$ *be linearly independent points of order* $2^m$, *and let* $\alpha \in \mathbb{Z}/2^m\mathbb{Z}$. *Then,*

$$\langle P + [\alpha]Q \rangle = \langle P + [\alpha](Q + [2^{m-1}]P) \rangle$$

*if and only if* $\alpha$ *is even.*

- This protocol is susceptible to an active attack when Alice reuses her key across multiple sessions.

**Lemma**

[*Galbraith, Petit, Shani, Ti*, 2016] *Let* $P, Q \in E[2^m]$ *be linearly independent points of order* $2^m$*, and let* $\alpha \in \mathbb{Z}/2^m\mathbb{Z}$*. Then,*

$$\langle P + [\alpha]Q \rangle = \langle P + [\alpha](Q + [2^{m-1}]P) \rangle$$

*if and only if* $\alpha$ *is even.*

- Suppose Alice and Bob verify they have the same shared secret.

- This protocol is susceptible to an active attack when Alice reuses her key across multiple sessions.

**Lemma**

[*Galbraith, Petit, Shani, Ti*, 2016] *Let* $P, Q \in E[2^m]$ *be linearly independent points of order* $2^m$*, and let* $\alpha \in \mathbb{Z}/2^m\mathbb{Z}$*. Then,*

$$\langle P + [\alpha]Q \rangle = \langle P + [\alpha](Q + [2^{m-1}]P) \rangle$$

*if and only if* $\alpha$ *is even.*

- Suppose Alice and Bob verify they have the same shared secret.
- Bob can be dishonest and use $(E_B, U, V + [2^{m-1}]U)$ as his public key.

- Alice will compute the elliptic curve $E_B/\langle U + [\alpha](V + [2^{m-1}]U)\rangle$.

- Alice will compute the elliptic curve $E_B/\langle U + [\alpha](V + [2^{m-1}]U)\rangle$.
- Bob can still compute the curve $E_A/\langle R + [\beta]S\rangle$.

- Alice will compute the elliptic curve $E_B/\langle U + [\alpha](V + [2^{m-1}]U)\rangle$.
- Bob can still compute the curve $E_A/\langle R + [\beta]S\rangle$.
- Bob learns the parity of $\alpha$ based on whether or not Alice computes the same curve.

- Alice will compute the elliptic curve $E_B/\langle U + [\alpha](V + [2^{m-1}]U)\rangle$.
- Bob can still compute the curve $E_A/\langle R + [\beta]S\rangle$.
- Bob learns the parity of $\alpha$ based on whether or not Alice computes the same curve.
- This attack can be extended adaptively to learn each bit of $\alpha$ efficiently

- Alice will compute the elliptic curve $E_B/\langle U + [\alpha](V + [2^{m-1}]U)\rangle$.
- Bob can still compute the curve $E_A/\langle R + [\beta]S\rangle$.
- Bob learns the parity of $\alpha$ based on whether or not Alice computes the same curve.
- This attack can be extended adaptively to learn each bit of $\alpha$ efficiently
- This modified public key is guaranteed to pass all known direct validation methods.

- Alice will compute the elliptic curve $E_B/\langle U + [\alpha](V + [2^{m-1}]U)\rangle$.
- Bob can still compute the curve $E_A/\langle R + [\beta]S\rangle$.
- Bob learns the parity of $\alpha$ based on whether or not Alice computes the same curve.
- This attack can be extended adaptively to learn each bit of $\alpha$ efficiently
- This modified public key is guaranteed to pass all known direct validation methods.
- This suggests that static keys can not be used for SIDH.

We let **KeyEst** be a key establishment function. A key agreement protocol, **KeyAgree**, for Alice and Bob consists of:

- **Setup**: Both members obtain a valid copy of the global parameters.

We let **KeyEst** be a key establishment function. A key agreement protocol, **KeyAgree**, for Alice and Bob consists of:

- **Setup**: Both members obtain a valid copy of the global parameters.

- **Key Generation**: Alice generates a secret key $s_A$ and public key $p_A$, likewise Bob generates $s_B$ and $p_B$.

We let **KeyEst** be a key establishment function. A key agreement protocol, **KeyAgree**, for Alice and Bob consists of:

- **Setup**: Both members obtain a valid copy of the global parameters.
- **Key Generation**: Alice generates a secret key $s_A$ and public key $p_A$, likewise Bob generates $s_B$ and $p_B$.
- **Communication**: Alice obtains $p_B$ and Bob obtains $p_A$.

We let **KeyEst** be a key establishment function. A key agreement protocol, **KeyAgree**, for Alice and Bob consists of:

- **Setup**: Both members obtain a valid copy of the global parameters.
- **Key Generation**: Alice generates a secret key $s_A$ and public key $p_A$, likewise Bob generates $s_B$ and $p_B$.
- **Communication**: Alice obtains $p_B$ and Bob obtains $p_A$.
- **Key Establishment**: Alice computes $K_A = \textbf{KeyEst}(p_B, s_A)$ and Bob computes $K_B = \textbf{KeyEst}(p_A, s_B)$.

We let **KeyEst** be a key establishment function. A key agreement protocol, **KeyAgree**, for Alice and Bob consists of:

- **Setup**: Both members obtain a valid copy of the global parameters.
- **Key Generation**: Alice generates a secret key $s_A$ and public key $p_A$, likewise Bob generates $s_B$ and $p_B$.
- **Communication**: Alice obtains $p_B$ and Bob obtains $p_A$.
- **Key Establishment**: Alice computes $K_A = \textbf{KeyEst}(p_B, s_A)$ and Bob computes $K_B = \textbf{KeyEst}(p_A, s_B)$.
- **Verification**: If applicable, each participant test the validity of the others public key. Alice and Bob verify that $K_A = K_B$ and terminate communication if any test fails.

- Alice and Bob interact twice, during communication and verification. We consider the attack model where Bob can act dishonestly in both phases.

- Alice and Bob interact twice, during communication and verification. We consider the attack model where Bob can act dishonestly in both phases.

- Bob can send a specially chosen public key, $p_B^*$, and send some $K_B^*$ during verification.

- Alice and Bob interact twice, during communication and verification. We consider the attack model where Bob can act dishonestly in both phases.
- Bob can send a specially chosen public key, $p_B^*$, and send some $K_B^*$ during verification.
- This attack gives Bob access to an oracle $\textbf{Oracle}_{\textbf{KeyAgree}}(p_B^*, K_B^*)$ which returns 1 if $K_B^* = \textbf{KeyEst}(p_B^*, s_A)$, and returns 0 otherwise.

- Alice and Bob interact twice, during communication and verification. We consider the attack model where Bob can act dishonestly in both phases.
- Bob can send a specially chosen public key, $p_B^*$, and send some $K_B^*$ during verification.
- This attack gives Bob access to an oracle $\textbf{Oracle}_{\textbf{KeyAgree}}(p_B^*, K_B^*)$ which returns 1 if $K_B^* = \textbf{KeyEst}(p_B^*, s_A)$, and returns 0 otherwise.
- When Bob acts dishonestly, this oracle may leak some information about Alice's secret key $s_A$. Our work will present such leakage.

Let $k$ be a positive integer, and let **H** be a preimage resistant hash function. The key agreement protocol between Alice and Bob, called $k-$ **KeyAgree**:

- **Setup**: Both members obtain a valid copy of the global parameters.

Let $k$ be a positive integer, and let **H** be a preimage resistant hash function. The key agreement protocol between Alice and Bob, called $k-$ **KeyAgree**:

- **Setup**: Both members obtain a valid copy of the global parameters.

- **Key Generation**: Alice generates $k$ secret key/public key pairs $(s_{Ai}, p_{Ai})$, $1 \leq i \leq k$. Likewise Bob generates $(s_{Bi}, p_{Bi})$ for $1 \leq i \leq k$.

Let $k$ be a positive integer, and let **H** be a preimage resistant hash function. The key agreement protocol between Alice and Bob, called $k -$ **KeyAgree**:

- **Setup**: Both members obtain a valid copy of the global parameters.
- **Key Generation**: Alice generates $k$ secret key/public key pairs $(s_{Ai}, p_{Ai})$, $1 \leq i \leq k$. Likewise Bob generates $(s_{Bi}, p_{Bi})$ for $1 \leq i \leq k$.
- **Communication**: Alice initiates communication and sends all $k$ of her public keys to Bob. Bob then sends all $k$ of his public keys to Alice.

- **Key Establishment**: Alice computes $z_{i,j} \leftarrow \mathbf{KeyEst}(p_{Bi}, s_{Aj})$ for every pair $1 \leq i, j \leq k$, then computes

$$K_A \leftarrow \mathbf{H}(z_{1,1}, \ldots, z_{1,k}, z_{2,1}, \ldots, z_{2,k}, \ldots, z_{k,1}, \ldots, z_{k,k}).$$

Similarly, Bob computes $z'_{i,j} \leftarrow \mathbf{KeyEst}(p_{Aj}, s_{Bi})$ for each pair $1 \leq i, j \leq k$, and then computes

$$K_B \leftarrow \mathbf{H}(z'_{1,1}, \ldots, z'_{1,k}, z'_{2,1}, \ldots, z'_{2,k}, \ldots, z'_{k,1}, \ldots, z'_{k,k}).$$

- **Key Establishment**: Alice computes $z_{i,j} \leftarrow \mathbf{KeyEst}(p_{Bi}, s_{Aj})$ for every pair $1 \leq i, j \leq k$, then computes

$$K_A \leftarrow \mathbf{H}(z_{1,1}, \ldots, z_{1,k}, z_{2,1}, \ldots, z_{2,k}, \ldots, z_{k,1}, \ldots, z_{k,k}).$$

  Similarly, Bob computes $z'_{i,j} \leftarrow \mathbf{KeyEst}(p_{Aj}, s_{Bi})$ for each pair $1 \leq i, j \leq k$, and then computes

$$K_B \leftarrow \mathbf{H}(z'_{1,1}, \ldots, z'_{1,k}, z'_{2,1}, \ldots, z'_{2,k}, \ldots, z'_{k,1}, \ldots, z'_{k,k}).$$

- **Verification**: If applicable, Alice and Bob test the validity of each others public keys. Alice and Bob verify that $K_A = K_B$. Either party terminates the session if their validity or verification tests fail.

| | | Bob | | | |
|---|---|---|---|---|---|
| | | $(p_{B1}, s_{B1})$ | $(p_{B2}, s_{B2})$ | $\cdots$ | $(p_{Bk}, s_{Bk})$ |
| Alice | $(p_{A1}, s_{A1})$ | $z_{1,1}$ | $z_{1,2}$ | $\cdots$ | $z_{1,k}$ |
| | $(p_{A2}, s_{A2})$ | $z_{2,1}$ | $z_{2,2}$ | $\cdots$ | $z_{2,k}$ |
| | $\vdots$ | $\vdots$ | $\vdots$ | $\ddots$ | $\vdots$ |
| | $(p_{Ak}, s_{Ak})$ | $z_{k,1}$ | $z_{k,2}$ | $\cdots$ | $z_{k,k}$ |

Figure: Honest $k$-Key Agreement

| | | Bob | | | |
|---|---|---|---|---|---|
| | | $(p_B^*, s_{B1})$ | $(p_{B2}, s_{B2})$ | $\cdots$ | $(p_{Bk}, s_{Bk})$ |
| Alice | $(p_{A1}, s_{A1})$ | $z_{1,1}$ | $z_{1,2}$ | $\cdots$ | $z_{1,k}$ |
| | $(p_{A2}, s_{A2})$ | $z_{2,1}$ | $z_{2,2}$ | $\cdots$ | $z_{2,k}$ |
| | $\vdots$ | $\vdots$ | $\vdots$ | $\ddots$ | $\vdots$ |
| | $(p_{Ak}, s_{Ak})$ | $z_{k,1}$ | $z_{k,2}$ | $\cdots$ | $z_{k,k}$ |

Figure: Dishonest $k$-Key Agreement

### Definition

If Bob has a public key/secret key pair $(p_B, s_B)$ for **KeyAgree** and is given two public keys $p_{A1}$ and $p_{A2}$ (derived from some secret keys $s_{A1}$, $s_{A2}$ which are unknown to Bob). A modified public key $p_B^*$ is called **malicious** if:

- $p_B^*$ passes all validation tests Alice performs in the verification phase,

### Definition

If Bob has a public key/secret key pair $(p_B, s_B)$ for **KeyAgree** and is given two public keys $p_{A1}$ and $p_{A2}$ (derived from some secret keys $s_{A1}$, $s_{A2}$ which are unknown to Bob). A modified public key $p_B^*$ is called **malicious** if:

- $p_B^*$ passes all validation tests Alice performs in the verification phase,
- $\mathbf{KeyEst}(p_B^*, s_{A1}) = \mathbf{KeyEst}(p_B, s_{A1})$,

### Definition

If Bob has a public key/secret key pair $(p_B, s_B)$ for **KeyAgree** and is given two public keys $p_{A1}$ and $p_{A2}$ (derived from some secret keys $s_{A1}$, $s_{A2}$ which are unknown to Bob). A modified public key $p_B^*$ is called **malicious** if:

- $p_B^*$ passes all validation tests Alice performs in the verification phase,
- $\textbf{KeyEst}(p_B^*, s_{A1}) = \textbf{KeyEst}(p_B, s_{A1})$,
- $\textbf{KeyEst}(p_B^*, s_{A2}) = \textbf{KeyEst}(p_B, s_{A2})$, and

### Definition

If Bob has a public key/secret key pair $(p_B, s_B)$ for **KeyAgree** and is given two public keys $p_{A1}$ and $p_{A2}$ (derived from some secret keys $s_{A1}$, $s_{A2}$ which are unknown to Bob). A modified public key $p_B^*$ is called **malicious** if:

- $p_B^*$ passes all validation tests Alice performs in the verification phase,
- $\mathbf{KeyEst}(p_B^*, s_{A1}) = \mathbf{KeyEst}(p_B, s_{A1})$,
- $\mathbf{KeyEst}(p_B^*, s_{A2}) = \mathbf{KeyEst}(p_B, s_{A2})$, and
- repsonses from $\mathbf{Oracle}_{\mathbf{KeyAgree}}(p_B^*, \cdot)$ leak information about Alice's private keys with non-negligible probability.

### Theorem

*Let* **KeyAgree** *be a key agreement protocol using* **KeyEst** *in which it is computationally infeasible to create a malicious public key. Let* $p_B^*$ *be a modified public key that passes all validity tests of* **KeyAgree** *and may leak information about private keys.*

*Suppose that in k-*$\mathbf{KeyAgree}$ *one of the k parts to Bob's public key is* $p_B^*$*. Associated to the equation* $\mathbf{Oracle}_{\mathbf{KeyAgree}}(p_B^*, \cdot) = 1$ *is a probability distribution among all dishonest shared secrets. Let* $\rho$ *denote the largest such probability.*

*Then for all choices of* $K_B^*$ *for* $k - \mathbf{KeyAgree}$ *the equation* $\mathbf{Oracle}_{k-\mathbf{KeyAgree}}(p_B^*, K_B^*) = 1$ *holds with probability bounded above by* $\rho^{k-1}$*.*

## Definition

Let $p = 2^m 3^n f \pm 1$ be prime and $E$ a supersingular elliptic curve over $\mathbb{F}_{p^2}$. Let $\ell^r$ be $2^m$ or $3^n$, and $P, Q \in E(\mathbb{F}_{p^2})$ be such that $\langle P, Q \rangle = E[\ell^r]$.

Given an elliptic curve $E'$ defined over $\mathbb{F}_{p^2}$ which is isogenous to $E$ of degree $\ell^r$, the **Supersingular Isogeny (SSI)** problem is to find an isogeny over $\mathbb{F}_{p^2}$ from $E$ to $E'$ of the degree $\ell^r$.

### Definition

Let $p = 2^m 3^n f \pm 1$ be prime and $E$ a supersingular elliptic curve over $\mathbb{F}_{p^2}$. Let $\ell^r$ be $2^m$ or $3^n$, and $P, Q \in E(\mathbb{F}_{p^2})$ be such that $\langle P, Q \rangle = E[\ell^r]$.

Given an elliptic curve $E'$ defined over $\mathbb{F}_{p^2}$ which is isogenous to $E$ of degree $\ell^r$, the **Supersingular Isogeny (SSI)** problem is to find an isogeny over $\mathbb{F}_{p^2}$ from $E$ to $E'$ of the degree $\ell^r$.

### Theorem

*Under the assumption that the SSI problem is intractable, it is computationally infeasible to find a malicious public key for SIDH with non-negligible probability.*

- **Classical Security:** The expected number of hashes before Bob determines the first bit of each of Alice's $k$ secret keys is

$$\sum_{i=0}^{\frac{k-2}{2}} \binom{k-1}{i} (\ell(\ell+1))^i.$$

To achieve $2^{128}$: $k = 60$ when $\ell = 2$, and $k = 50$ when $\ell = 3$.

- **Classical Security:** The expected number of hashes before Bob determines the first bit of each of Alice's $k$ secret keys is

$$\sum_{i=0}^{\frac{k-2}{2}} \binom{k-1}{i} (\ell(\ell+1))^i.$$

  To achieve $2^{128}$: $k = 60$ when $\ell = 2$, and $k = 50$ when $\ell = 3$.

- **Quantum Security:** Applying Grover's algorithm over a non-uniform search space results in $2^{128}$ operations when $\ell = 2, k = 92$, and $\ell = 3, k = 70$.

- **Classical Security:** The expected number of hashes before Bob determines the first bit of each of Alice's $k$ secret keys is

$$\sum_{i=0}^{\frac{k-2}{2}} \binom{k-1}{i} (\ell(\ell+1))^i.$$

  To achieve $2^{128}$: $k = 60$ when $\ell = 2$, and $k = 50$ when $\ell = 3$.

- **Quantum Security:** Applying Grover's algorithm over a non-uniform search space results in $2^{128}$ operations when $\ell = 2, k = 92$, and $\ell = 3, k = 70$.

- **Key Sizes:** An SIDH public key can be represented in 331 bytes at the 128-bit quantum security level. A $k$-SIDH pubic key requires $331 \times k$ bytes, or 31 kb at the same level.

- Security against specific active attacks grow exponentially in $k$, key size increases by a factor of $k$, and computation cost increases by a factor of $k^2$ for each participant.

- Security against specific active attacks grow exponentially in $k$, key size increases by a factor of $k$, and computation cost increases by a factor of $k^2$ for each participant.

- No other currently known post-quantum scheme achieves secure static-static key agreement.

- Security against specific active attacks grow exponentially in $k$, key size increases by a factor of $k$, and computation cost increases by a factor of $k^2$ for each participant.
- No other currently known post-quantum scheme achieves secure static-static key agreement.
- Transformation for key agreement protocols to protect against such active attacks which use verification.

- Security against specific active attacks grow exponentially in $k$, key size increases by a factor of $k$, and computation cost increases by a factor of $k^2$ for each participant.
- No other currently known post-quantum scheme achieves secure static-static key agreement.
- Transformation for key agreement protocols to protect against such active attacks which use verification.
- Future work includes reducing the quadratic cost, using economies of scale to optimize implementations, reducing $k$, determining the generality of our transformation requirement.

- Security against specific active attacks grow exponentially in $k$, key size increases by a factor of $k$, and computation cost increases by a factor of $k^2$ for each participant.
- No other currently known post-quantum scheme achieves secure static-static key agreement.
- Transformation for key agreement protocols to protect against such active attacks which use verification.
- Future work includes reducing the quadratic cost, using economies of scale to optimize implementations, reducing $k$, determining the generality of our transformation requirement.
- Thank you.