Key Compression for Isogeny-Based Cryptosystems

R. Azarderakhsh, D. Jao, K. Kalach, B. Koziel, and C. Leonardi

Department of Combinatorics & Optimization
Centre for Applied Cryptographic Research

UNIVERSITY OF WATERLOO

May 30, 2016

- We show how to compress public keys in the isogeny-based cryptosystem of Luca De Feo and David Jao to half the original size, with no change to the security level.

- We show how to compress public keys in the isogeny-based cryptosystem of Luca De Feo and David Jao to half the original size, with no change to the security level.

- We give a comparison of key sizes at fixed security levels between all known quantum-safe public key encryption schemes to show that isogenies achieve the smallest key sizes when compressed.

- An elliptic curve $E : y^2 = x^3 + ax + b$, defined over the finite field $\mathbb{F}_{p^n}$, is the set

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

  which has an additive group structure where the identity is the point ∞.

- An elliptic curve $E : y^2 = x^3 + ax + b$, defined over the finite field $\mathbb{F}_{p^n}$, is the set

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

  which has an additive group structure where the identity is the point ∞.

- For an integer m, we can define the torsion subgroup $E[m] = \{P \in E(\mathbb{F}_{p^n}) : mP = \infty\}$.

- An elliptic curve $E : y^2 = x^3 + ax + b$, defined over the finite field $\mathbb{F}_{p^n}$, is the set

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

  which has an additive group structure where the identity is the point ∞.

- For an integer m, we can define the torsion subgroup $E[m] = \{P \in E(\mathbb{F}_{p^n}) : mP = \infty\}$.

- A $\mathbb{Z}$-basis for the subgroup $E[m]$ is a set $\{P_1, \ldots, P_r\}$ such that $E[m] = \{\sum_{i=1}^{r} a_i P_i : a_i \in \mathbb{Z}\}$, and the P_i's are linearly independent.

- For an elliptic curve $E(\mathbb{F}_{p^n})$, define the j-invariant as

$$j(E) = 1728\frac{4a^3}{4a^3 + 27b^2}.$$

- For an elliptic curve $E(\mathbb{F}_{p^n})$, define the j-invariant as

$$j(E) = 1728\frac{4a^3}{4a^3 + 27b^2}.$$

- $j(E) = j(E')$ if and only if $E(\overline{\mathbb{F}}_{p^n})$ is isomorphic to $E'(\overline{\mathbb{F}}_{p^n})$.

- An isogeny $\phi : E(\mathbb{F}_{p^n}) \rightarrow E'(\mathbb{F}_{p^n})$ is a non-constant group homomorphism satisfying $\phi(\infty) = \infty$.

- An isogeny $\phi : E(\mathbb{F}_{p^n}) \to E'(\mathbb{F}_{p^n})$ is a non-constant group homomorphism satisfying $\phi(\infty) = \infty$.

- Define the *kernel* of an isogeny to be the set $\{P \in E(\mathbb{F}_{p^n}) : \phi(P) = \infty\}$.

- An isogeny $\phi : E(\mathbb{F}_{p^n}) \rightarrow E'(\mathbb{F}_{p^n})$ is a non-constant group homomorphism satisfying $\phi(\infty) = \infty$.

- Define the *kernel* of an isogeny to be the set $\{P \in E(\mathbb{F}_{p^n}) : \phi(P) = \infty\}$.

- For every subgroup G of and elliptic curve $E(\mathbb{F}_{p^n})$, there exists a unique (separable) isogeny $\phi : E(\mathbb{F}_{p^n}) \rightarrow E'(\mathbb{F}_{p^n})$ up to isomorphism such that $\ker\phi = G$.

The Jao-De Feo encryption scheme (2011):

- Public parameters: $p = 2^{e_2}3^{e_3}f - 1$, a prime.

The Jao-De Feo encryption scheme (2011):

- Public parameters: $p = 2^{e_2}3^{e_3}f - 1$, a prime.
- $E : y^2 = x^3 + ax + b$ defined over $\mathbb{F}_{p^2}$, having $(p+1)^2 = (2^{e_2}3^{e_3}f)^2$ points. Elliptic curves with this particular number of points are called supersingular.

The Jao-De Feo encryption scheme (2011):

- Public parameters: $p = 2^{e_2}3^{e_3}f - 1$, a prime.
- $E : y^2 = x^3 + ax + b$ defined over $\mathbb{F}_{p^2}$, having $(p+1)^2 = (2^{e_2}3^{e_3}f)^2$ points. Elliptic curves with this particular number of points are called supersingular.
- A $\mathbb{Z}$-basis $\{P_A, Q_A\}$ for the subgroup

$$E[2^{e_2}] = \{P \in E(\mathbb{F}_{p^2}) : 2^{e_2}P = \infty\}$$

The Jao-De Feo encryption scheme (2011):

- Public parameters: $p = 2^{e_2}3^{e_3}f - 1$, a prime.
- $E : y^2 = x^3 + ax + b$ defined over $\mathbb{F}_{p^2}$, having $(p+1)^2 = (2^{e_2}3^{e_3}f)^2$ points. Elliptic curves with this particular number of points are called supersingular.
- A $\mathbb{Z}$-basis $\{P_A, Q_A\}$ for the subgroup

$$E[2^{e_2}] = \{P \in E(\mathbb{F}_{p^2}) : 2^{e_2}P = \infty\}$$

- A $\mathbb{Z}$-basis $\{P_B, Q_B\}$ for the subgroup

$$E[3^{e_3}] = \{P \in E(\mathbb{F}_{p^2}) : 3^{e_3}P = \infty\}$$

- Alice: Choose a random point $R_A = m_A P_A + n_A Q_A$ of order 2^{e_2}.
  Compute $\phi_A : E \to E/\langle R_A \rangle$.
  Send $E/\langle R_A \rangle, \phi_A(P_B), \phi_A(Q_B)$ to Bob.

- Alice: Choose a random point $R_A = m_A P_A + n_A Q_A$ of order 2^{e_2}.
  Compute $\phi_A : E \to E/\langle R_A \rangle$.
  Send $E/\langle R_A \rangle, \phi_A(P_B), \phi_A(Q_B)$ to Bob.

- Bob: Choose a random point $R_B = m_B P_B + n_B Q_B$ of order 3^{e_3}.
  Compute $\phi_B : E \to E/\langle R_B \rangle$.
  Send $E/\langle R_B \rangle, \phi_B(P_A), \phi_B(Q_A)$ to Alice.

- Alice: Choose a random point $R_A = m_A P_A + n_A Q_A$ of order 2^{e_2}.
  Compute $\phi_A : E \to E/\langle R_A \rangle$.
  Send $E/\langle R_A \rangle, \phi_A(P_B), \phi_A(Q_B)$ to Bob.

- Bob: Choose a random point $R_B = m_B P_B + n_B Q_B$ of order 3^{e_3}.
  Compute $\phi_B : E \to E/\langle R_B \rangle$.
  Send $E/\langle R_B \rangle, \phi_B(P_A), \phi_B(Q_A)$ to Alice.

- Alice and Bob now share the secret:

$$E/\langle R_A, R_B \rangle = (E/\langle R_A \rangle)/\langle m_B \phi_A(P_B) + n_B \phi_A(Q_B) \rangle$$

$$= (E/\langle R_B \rangle)/\langle m_A \phi_B(P_A) + n_A \phi_B(Q_A) \rangle$$

- Alice: Choose a random point $R_A = m_A P_A + n_A Q_A$ of order 2^{e_2}.
  Compute $\phi_A : E \to E/\langle R_A \rangle$.
  Send $E/\langle R_A \rangle, \phi_A(P_B), \phi_A(Q_B)$ to Bob.

- Bob: Choose a random point $R_B = m_B P_B + n_B Q_B$ of order 3^{e_3}.
  Compute $\phi_B : E \to E/\langle R_B \rangle$.
  Send $E/\langle R_B \rangle, \phi_B(P_A), \phi_B(Q_A)$ to Alice.

- Alice and Bob now share the secret:

$$E/\langle R_A, R_B \rangle = (E/\langle R_A \rangle)/\langle m_B \phi_A(P_B) + n_B \phi_A(Q_B) \rangle$$
$$= (E/\langle R_B \rangle)/\langle m_A \phi_B(P_A) + n_A \phi_B(Q_A) \rangle$$

- The shared secret $E/\langle R_A, R_B \rangle$ is only defined up to isomorphism, so instead of using the curve we use the j-invariant of $E/\langle R_A, R_B \rangle$.

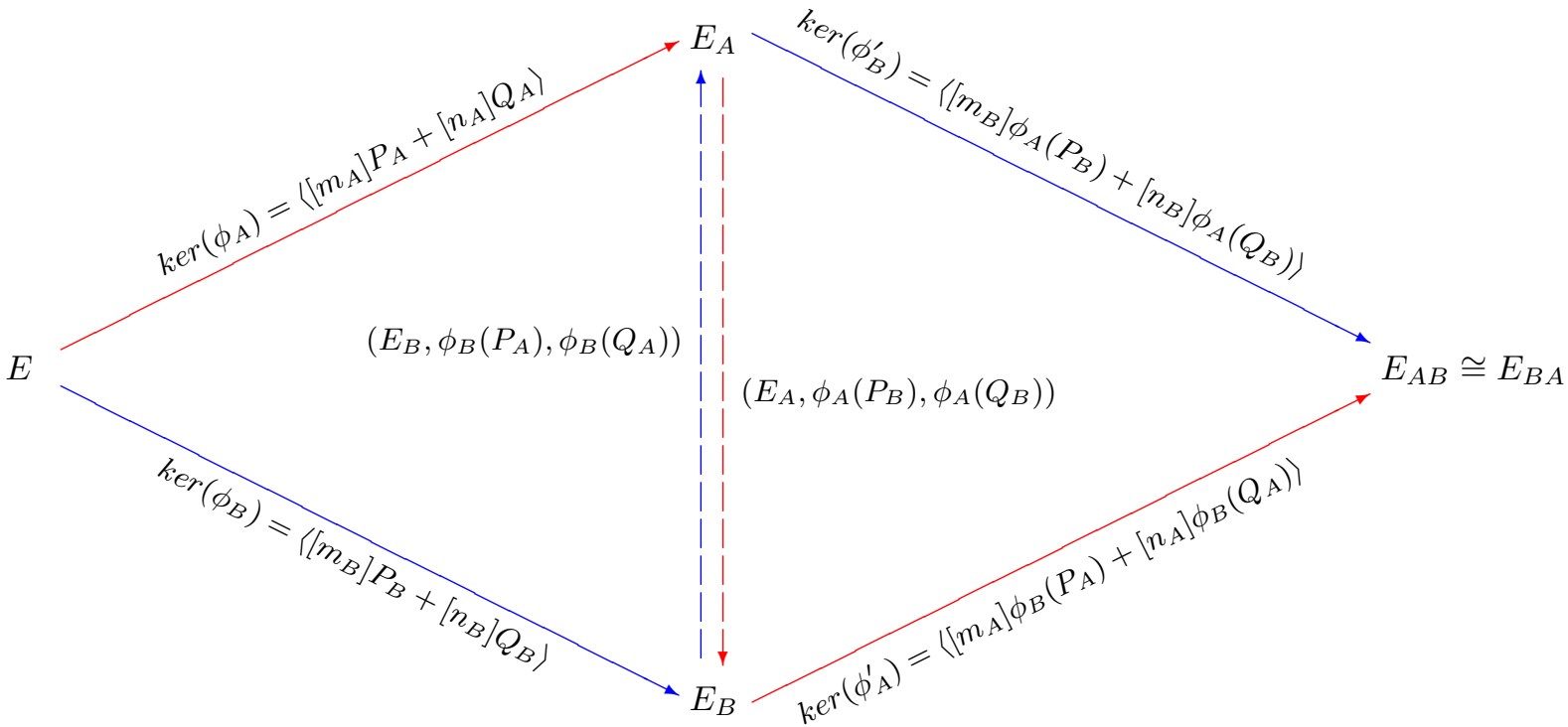

Fig. 1: Isogeny-based key exchange protocol

- The most efficient known quantum algorithm for the underlying isogeny computation problem has runtime $O(p^{1/6})$.

- The most efficient known quantum algorithm for the underlying isogeny computation problem has runtime $O(p^{1/6})$.

- To achieve a k-bit security level (against a quantum adversary) $p = 2^{e_2}3^{e_3} \cdot f - 1$ must be approximately $6k$-bits. Then each of 2^{e_2} and 3^{e_3} need to have $3k$ bits. For example, for 128-bit security, take $p \approx 2^{6 \cdot 128} = 2^{768}$.

- The most efficient known quantum algorithm for the underlying isogeny computation problem has runtime $O(p^{1/6})$.

- To achieve a k-bit security level (against a quantum adversary) $p = 2^{e_2}3^{e_3} \cdot f - 1$ must be approximately $6k$-bits. Then each of 2^{e_2} and 3^{e_3} need to have $3k$ bits. For example, for 128-bit security, take $p \approx 2^{6 \cdot 128} = 2^{768}$.

- Alice's public key is $a, b \in \mathbb{F}_{p^2}$ ($12k$ bits each) and $\phi_A(P_B), \phi_A(Q_B) \in E_A(\mathbb{F}_{p^2})$ ($12k$ bits for each x-coordinate), giving a total of $48k$ bits required to achieve a k-bit level of security.

- Both $\phi_A(P_B)$ and $\phi_A(Q_B)$ have order 3^{e_3}, which means they are elements of the subgroup $E_A[3^{e_3}]$.

- Both $\phi_A(P_B)$ and $\phi_A(Q_B)$ have order 3^{e_3}, which means they are elements of the subgroup $E_A[3^{e_3}]$.

- If Alice and Bob could compute the same basis $\{B_1, B_2\}$ for $E_A[3^{e_3}]$ then they could use integers in $\mathbb{Z}/3^{e_3}\mathbb{Z}$ to represent $\phi_A(P_B)$ and $\phi_A(Q_B)$ as linear combinations of B_1 and B_2:

$$\phi_A(P_B) = \alpha_1 B_1 + \beta_1 B_2, \text{ and } \phi_A(Q_B) = \alpha_2 B_1 + \beta_2 B_2$$

- Both $\phi_A(P_B)$ and $\phi_A(Q_B)$ have order 3^{e_3}, which means they are elements of the subgroup $E_A[3^{e_3}]$.

- If Alice and Bob could compute the same basis $\{B_1, B_2\}$ for $E_A[3^{e_3}]$ then they could use integers in $\mathbb{Z}/3^{e_3}\mathbb{Z}$ to represent $\phi_A(P_B)$ and $\phi_A(Q_B)$ as linear combinations of B_1 and B_2:

$$\phi_A(P_B) = \alpha_1 B_1 + \beta_1 B_2, \text{ and } \phi_A(Q_B) = \alpha_2 B_1 + \beta_2 B_2$$

- Each integer in $\mathbb{Z}/3^{e_3}\mathbb{Z}$ is approximately $\frac{1}{2}p = 3k$. The combined size of $\alpha_1, \beta_1, \alpha_2, \beta_2$ is $12k$ bits. This method reduces the bits required to compute points by half.

Implementation details:

- Fix a hash function $H : \mathbb{Z} \to E_A$.

Implementation details:

- Fix a hash function $H : \mathbb{Z} \to E_A$.
- Compute $H(i), i = 0, 1, 2, \ldots$ until a point, $H(i_0)$ is found whose order is a multiple of 3^{e_3}.

Implementation details:

- Fix a hash function $H : \mathbb{Z} \to E_A$.
- Compute $H(i), i = 0, 1, 2, \ldots$ until a point, $H(i_0)$ is found whose order is a multiple of 3^{e_3}.
- Multiply $H(i_0)$ by an appropriate scaler to make its order exactly 3^{e_3}. Set this point to B_1.

Implementation details:

- Fix a hash function $H : \mathbb{Z} \to E_A$.
- Compute $H(i), i = 0, 1, 2, \ldots$ until a point, $H(i_0)$ is found whose order is a multiple of 3^{e_3}.
- Multiply $H(i_0)$ by an appropriate scaler to make its order exactly 3^{e_3}. Set this point to B_1.
- Compute $H(j), j = i_0 + 1, i_0 + 2, \ldots$ until a point, $H(j_0)$, is found which is not a multiple of B_1 and whose order is a multiple of 3^{e_3}.

Implementation details:

- Fix a hash function $H : \mathbb{Z} \to E_A$.
- Compute $H(i), i = 0, 1, 2, \ldots$ until a point, $H(i_0)$ is found whose order is a multiple of 3^{e_3}.
- Multiply $H(i_0)$ by an appropriate scaler to make its order exactly 3^{e_3}. Set this point to B_1.
- Compute $H(j), j = i_0 + 1, i_0 + 2, \ldots$ until a point, $H(j_0)$, is found which is not a multiple of B_1 and whose order is a multiple of 3^{e_3}.
- Multiply $H(j_0)$ by an appropriate scalar to make its order exactly 3^{e_3}. Set this point to B_2 and return the basis $\{B_1, B_2\}$ for $E_A[3^{e_3}]$.

The final part of point compression requires Alice to compute integers α, β such that $R = \alpha B_1 + \beta B_2$, for a given basis $\{B_1, B_2\}$ and a given point R. This two-dimensional discrete logarithm can be turned into two instances of the standard discrete log problem (which is easy to solve in $E_A[3^{e_3}]$).

- Compute the Weil pairing

$$e(B_1, R) = e(B_1, \alpha B_1 + \beta B_2) = e(B_1, B_1)^{\alpha} e(B_1, B_2)^{\beta} = e(B_1, B_2)^{\beta}$$

The final part of point compression requires Alice to compute integers α, β such that $R = \alpha B_1 + \beta B_2$, for a given basis $\{B_1, B_2\}$ and a given point R. This two-dimensional discrete logarithm can be turned into two instances of the standard discrete log problem (which is easy to solve in $E_A[3^{e_3}]$).

- Compute the Weil pairing

$$e(B_1, R) = e(B_1, \alpha B_1 + \beta B_2) = e(B_1, B_1)^{\alpha} e(B_1, B_2)^{\beta} = e(B_1, B_2)^{\beta}$$

- Compute the discrete log of $e(B_1, R)$ with base $e(B_1, B_2)$ to determine β.

The final part of point compression requires Alice to compute integers α, β such that $R = \alpha B_1 + \beta B_2$, for a given basis $\{B_1, B_2\}$ and a given point R. This two-dimensional discrete logarithm can be turned into two instances of the standard discrete log problem (which is easy to solve in $E_A[3^{e_3}]$).

- Compute the Weil pairing

$$e(B_1, R) = e(B_1, \alpha B_1 + \beta B_2) = e(B_1, B_1)^{\alpha} e(B_1, B_2)^{\beta} = e(B_1, B_2)^{\beta}$$

- Compute the discrete log of $e(B_1, R)$ with base $e(B_1, B_2)$ to determine β.
- Similarly, α is the discrete log of $e(B_2, R)$ with base $e(B_2, B_1)$.

Transmitting points $\phi_A(P_B), \phi_A(Q_B)$ as basis coefficients:

- Compute the basis $\{B_1, B_2\}$ using the same hash function as Bob.

Transmitting points $\phi_A(P_B), \phi_A(Q_B)$ as basis coefficients:

- Compute the basis $\{B_1, B_2\}$ using the same hash function as Bob.

- Compute $\alpha_1, \beta_1, \alpha_2, \beta_2$ using Weil pairings, such that $\phi_A(P_B) = \alpha_1 B_1 + \beta_1 B_2$, and $\phi_A(Q_B) = \alpha_2 B_1 + \beta_2 B_2$.

Transmitting points $\phi_A(P_B), \phi_A(Q_B)$ as basis coefficients:

- Compute the basis $\{B_1, B_2\}$ using the same hash function as Bob.
- Compute $\alpha_1, \beta_1, \alpha_2, \beta_2$ using Weil pairings, such that $\phi_A(P_B) = \alpha_1 B_1 + \beta_1 B_2$, and $\phi_A(Q_B) = \alpha_2 B_1 + \beta_2 B_2$.
- Send $\{\alpha_1, \beta_1, \alpha_2, \beta_2\}$ to Bob.

- Since the shared secret $j(E/\langle R_A, R_B \rangle)$ is only defined up to isomorphism, it is sufficient to only define E_A up to isomorphism as well.

- Since the shared secret $j(E/\langle R_A, R_B \rangle)$ is only defined up to isomorphism, it is sufficient to only define E_A up to isomorphism as well.

- Instead of sending the equation coefficients $a, b \in \mathbb{F}_{p^2}$ of $E_A : y^2 = x^3 + ax + b$, Alice can send $j(E_A) \in \mathbb{F}_{p^2}$ This requires sending only one element of $\mathbb{F}_{p^2}$ instead of two, again reducing the size by a factor of two.

- Bob must reconstruct the curve E_A from a given $j(E_A)$, or any curve E_A^* in the same isomorphism class. There are standard formulas for this procedure.

- Bob must reconstruct the curve E_A from a given $j(E_A)$, or any curve E_A^* in the same isomorphism class. There are standard formulas for this procedure.

- Bob must also reconstruct the points $\phi_A(P_B), \phi_A(Q_B)$, but these points are on E_A, not E_A^*. Alice must anticipate this issue and compensate for this by mapping $\phi_A(P_B), \phi_A(Q_B)$ onto E_A^* before the point decompression algorithm.

Given a public key $(E_A, \phi_A(P_B), \phi_A(Q_B))$:

- Compute $j(E_A)$.

Given a public key $(E_A, \phi_A(P_B), \phi_A(Q_B))$:

- Compute $j(E_A)$.
- Compute E_A^* from $j(E_A)$.

Given a public key $(E_A, \phi_A(P_B), \phi_A(Q_B))$:

- Compute $j(E_A)$.
- Compute E_A^* from $j(E_A)$.
- Compute the basis $\{B_1, B_2\}$ for $E_A^*[3^{e_3}]$.

Given a public key $(E_A, \phi_A(P_B), \phi_A(Q_B))$:

- Compute $j(E_A)$.
- Compute E_A^* from $j(E_A)$.
- Compute the basis $\{B_1, B_2\}$ for $E_A^*[3^{e_3}]$.
- Compute the image, R_1, of $\phi_A(P_B)$ under the isomorphism $E_A \simeq E_A^*$.

Given a public key $(E_A, \phi_A(P_B), \phi_A(Q_B))$:

- Compute $j(E_A)$.
- Compute E_A^* from $j(E_A)$.
- Compute the basis $\{B_1, B_2\}$ for $E_A^*[3^{e_3}]$.
- Compute the image, R_1, of $\phi_A(P_B)$ under the isomorphism $E_A \simeq E_A^*$.
- Compute the image, R_2, of $\phi_A(Q_B)$ under the isomorphism $E_A \simeq E_A^*$.

Given a public key $(E_A, \phi_A(P_B), \phi_A(Q_B))$:

- Compute $j(E_A)$.
- Compute E_A^* from $j(E_A)$.
- Compute the basis $\{B_1, B_2\}$ for $E_A^*[3^{e_3}]$.
- Compute the image, R_1, of $\phi_A(P_B)$ under the isomorphism $E_A \simeq E_A^*$.
- Compute the image, R_2, of $\phi_A(Q_B)$ under the isomorphism $E_A \simeq E_A^*$.
- Compute the coefficients $\alpha_1, \beta_1 \in \mathbb{Z}/3^{e_3}\mathbb{Z}$ such that $R_1 = \alpha_1 B_1 + \beta_1 B_2$.

Given a public key $(E_A, \phi_A(P_B), \phi_A(Q_B))$:

- Compute $j(E_A)$.
- Compute E_A^* from $j(E_A)$.
- Compute the basis $\{B_1, B_2\}$ for $E_A^*[3^{e_3}]$.
- Compute the image, R_1, of $\phi_A(P_B)$ under the isomorphism $E_A \simeq E_A^*$.
- Compute the image, R_2, of $\phi_A(Q_B)$ under the isomorphism $E_A \simeq E_A^*$.
- Compute the coefficients $\alpha_1, \beta_1 \in \mathbb{Z}/3^{e_3}\mathbb{Z}$ such that $R_1 = \alpha_1 B_1 + \beta_1 B_2$.
- Compute the coefficients $\alpha_2, \beta_2 \in \mathbb{Z}/3^{e_3}\mathbb{Z}$ such that $R_2 = \alpha_2 B_1 + \beta_2 B_2$.

Given a public key $(E_A, \phi_A(P_B), \phi_A(Q_B))$:

- Compute $j(E_A)$.
- Compute E_A^* from $j(E_A)$.
- Compute the basis $\{B_1, B_2\}$ for $E_A^*[3^{e_3}]$.
- Compute the image, R_1, of $\phi_A(P_B)$ under the isomorphism $E_A \simeq E_A^*$.
- Compute the image, R_2, of $\phi_A(Q_B)$ under the isomorphism $E_A \simeq E_A^*$.
- Compute the coefficients $\alpha_1, \beta_1 \in \mathbb{Z}/3^{e_3}\mathbb{Z}$ such that $R_1 = \alpha_1 B_1 + \beta_1 B_2$.
- Compute the coefficients $\alpha_2, \beta_2 \in \mathbb{Z}/3^{e_3}\mathbb{Z}$ such that $R_2 = \alpha_2 B_1 + \beta_2 B_2$.
- The compressed public key is $(j(E_A), \alpha_1, \beta_1, \alpha_2, \beta_2)$.

Given a compressed key $(j(E_A), \alpha_1, \beta_1, \alpha_2, \beta_2)$:

- Compute the curve E_A^* from $j(E_A)$.

Given a compressed key $(j(E_A), \alpha_1, \beta_1, \alpha_2, \beta_2)$:

- Compute the curve E_A^* from $j(E_A)$.
- Compute the basis $\{B_1, B_2\}$ for $E_A^*[3^{e_3}]$.

Given a compressed key $(j(E_A), \alpha_1, \beta_1, \alpha_2, \beta_2)$:

- Compute the curve E_A^* from $j(E_A)$.
- Compute the basis $\{B_1, B_2\}$ for $E_A^*[3^{e_3}]$.
- Compute $R_1 = \alpha_1 B_1 + \beta_1 B_2$.

Given a compressed key $(j(E_A), \alpha_1, \beta_1, \alpha_2, \beta_2)$:

- Compute the curve E_A^* from $j(E_A)$.
- Compute the basis $\{B_1, B_2\}$ for $E_A^*[3^{e_3}]$.
- Compute $R_1 = \alpha_1 B_1 + \beta_1 B_2$.
- Compute $R_2 = \alpha_2 B_1 + \beta_2 B_2$.

Given a compressed key $(j(E_A), \alpha_1, \beta_1, \alpha_2, \beta_2)$:

- Compute the curve E_A^* from $j(E_A)$.
- Compute the basis $\{B_1, B_2\}$ for $E_A^*[3^{e_3}]$.
- Compute $R_1 = \alpha_1 B_1 + \beta_1 B_2$.
- Compute $R_2 = \alpha_2 B_1 + \beta_2 B_2$.
- The uncompressed key is (E_A^*, R_1, R_2).

Given a compressed key $(j(E_A), \alpha_1, \beta_1, \alpha_2, \beta_2)$:

- Compute the curve E_A^* from $j(E_A)$.
- Compute the basis $\{B_1, B_2\}$ for $E_A^*[3^{e_3}]$.
- Compute $R_1 = \alpha_1 B_1 + \beta_1 B_2$.
- Compute $R_2 = \alpha_2 B_1 + \beta_2 B_2$.
- The uncompressed key is (E_A^*, R_1, R_2).

- For the k-bit level of security, we have now reduced the size of a public key from $48k$ bits to $24k$ bits.

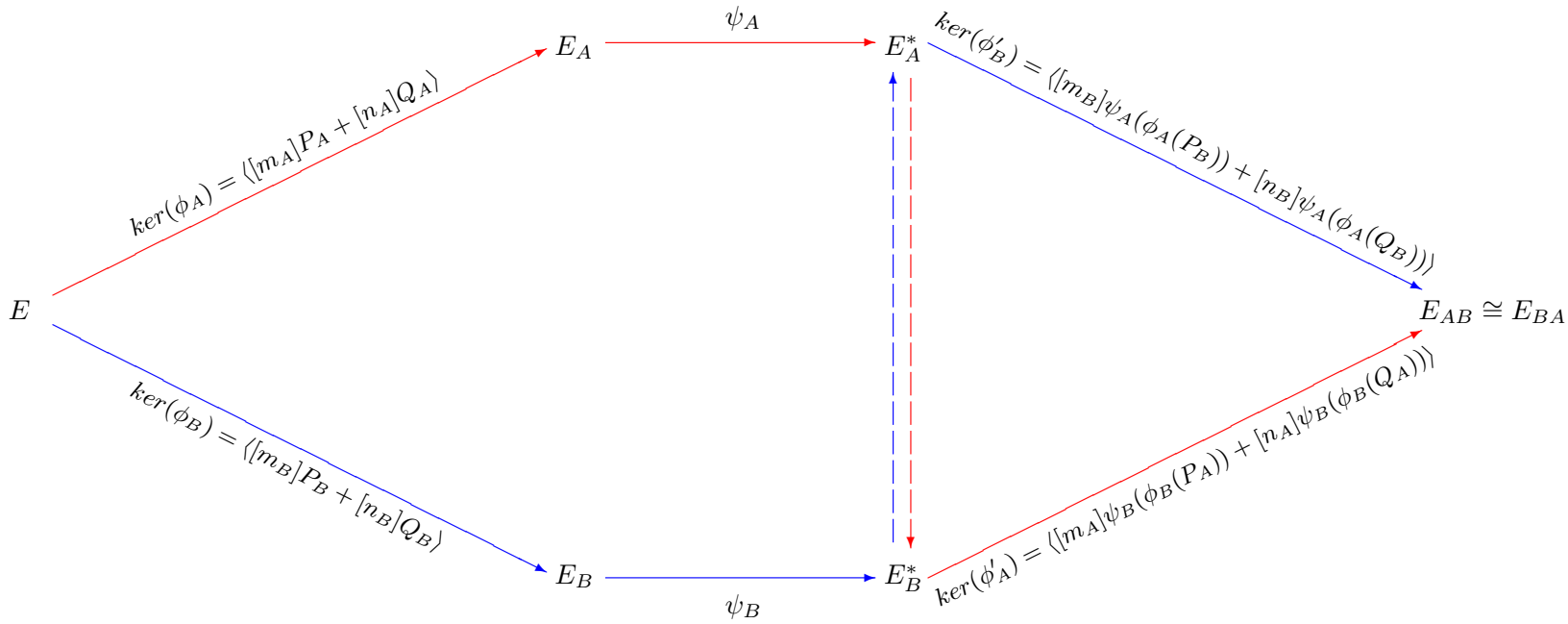

Fig. 1: Key exchange protocol with compression

Known families of post-quantum cryptosystems which support encryption:

- Lattice-based schemes: NTRU, Ring-LWE

Known families of post-quantum cryptosystems which support encryption:

- Lattice-based schemes: NTRU, Ring-LWE
- Code-based schemes: McEliece, Niederreiter

Known families of post-quantum cryptosystems which support encryption:

- Lattice-based schemes: NTRU, Ring-LWE
- Code-based schemes: McEliece, Niederreiter
- Isogenies

Known families of post-quantum cryptosystems which support encryption:

- Lattice-based schemes: NTRU, Ring-LWE
- Code-based schemes: McEliece, Niederreiter
- Isogenies
- Other post-quantum cryptosystems include hash-based and multivariate polynomials, but they support signatures.

- Isogeny: [Jao and De Feo., PQCrypto 2011]

Security level	Public key (bits)
112	~~5 378~~ 2 689
128	~~6 146~~ 3 073
192	~~9 218~~ 4 609
256	~~12 290~~ 6 145

- Isogeny: [Jao and De Feo., PQCrypto 2011]

Security level	Public key (bits)
112	~~5 378~~ 2 689
128	~~6 146~~ 3 073
192	~~9 218~~ 4 609
256	~~12 290~~ 6 145

- The isogeny-based public key size for 128-bit security is 384 bytes, which makes it the only post-quantum cryptosystem to fit within the 514 byte cell size for the popular software Tor.

128-bit security level:

Scheme	Public key (bits)
NTRU	4 939
Ring-LWE	7 498
McEliece (Goppa)	1 991 880
McEliece (QD-MDPC)	9 857
Isogenies	3 073

256-bit security level:

Scheme	Public key (bits)
NTRU	11 957
Ring-LWE	15 690
McEliece (Goppa)	9 276 241
McEliece (QD-MDPC)	32 771
Isogenies	6 145

Runtimes of our implementation for key-exchange in Magma

$p = 2^{e_A}3^{e_B}f \pm 1$	$2^{386}3^{242}2 - 1$	$2^{514}3^{323}353 - 1$
Classical security (bits)	192	256
Quantum security (bits)	128	168
Alice Compression (sec)	1.988	3.318
Bob Compression (sec)	1.403	2.404
Alice Decompression (sec)	0.530	1.020
Bob Decompression (sec)	0.650	1.100

- Compression and decompression require no private information (m_A, n_A were never used), and hence, have no impact on the security of the scheme.

- Compression and decompression require no private information (m_A, n_A were never used), and hence, have no impact on the security of the scheme.

- The computational costs of compression and decompression are incurred only once per key (compression is needed once per key for the lifetime of the key, and decompression is needed once per key per recipient).