Composable Security of Delegated Quantum Computation

Vedran Dunjko[1] Joseph Fitzsimons[2]
Christopher Portmann[3] Renato Renner[3]

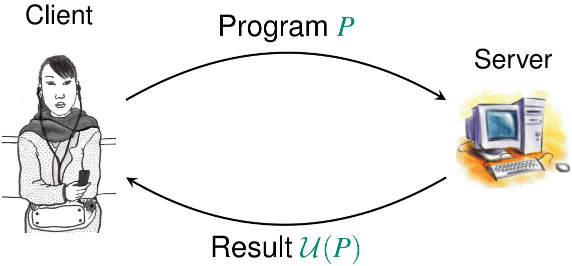
[1]University of Innsbruck

[2]National University of Singapore

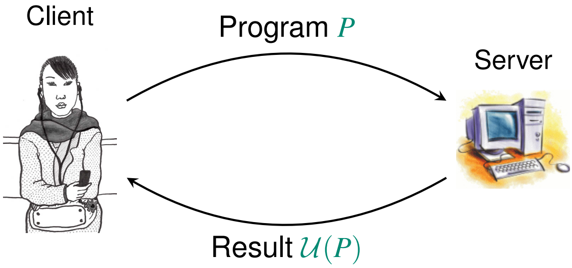
[3]ETH Zurich

ASIACRYPT, 11 December 2014

Delegated computation

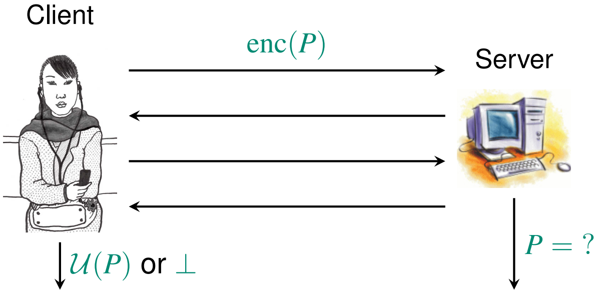


Delegated computation



We would like:

- Blindness: The server learns nothing about P.
- Verifiability: The client can check that the result is correct.



We would like:

- Blindness: The server learns nothing about P.
- Verifiability: The client can check that the result is correct.

Delegated computation

Client

$\mathrm{enc}(P)$

Server

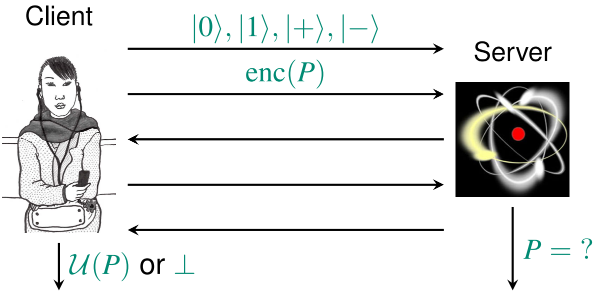
$\mathcal{U}(P)$ or $\perp$

$P = ?$

Impossibility result [Abadi, Feigenbaum, Kilian 1987]

To achieve this with information-theoretic security, the client's protocol must have the same runtime as the server.

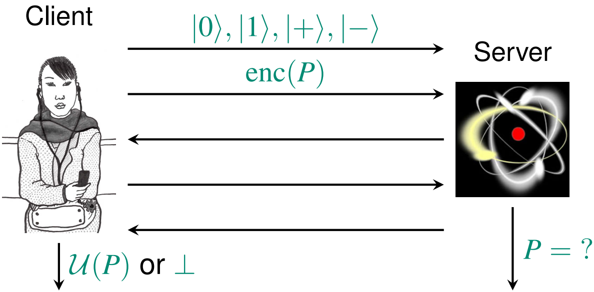
Delegated *quantum* computation



Observation

If the server is a universal quantum computer, but the client is not, the client can efficiently delegate an efficient quantum computation without violating the impossibility result.

Delegated *quantum* computation



Requirements on the client:

- Prepare and send 8 different single qubit states. [Broadbent, Fitzsimons, Kashefi 2009; FK 2012]
- Perform single qubit measurements. [Morimae, Fujii 2013; M 2014]

Ad hoc security definitions

Blindness (informal)

The server S obtains (approximately) no information about the program P, i.e.,

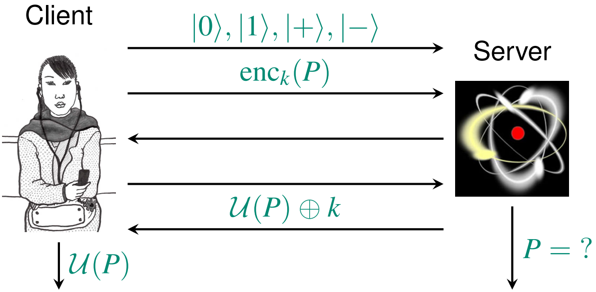
$$H(P|S) \approx_\varepsilon H(P).$$

Verifiability (informal)

(With high probability) the client does not accept a wrong result, i.e.,

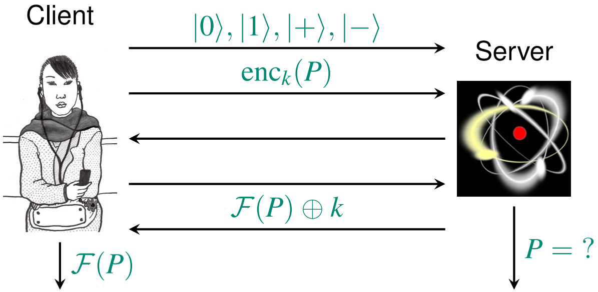
$$\Pr[\text{Output} = \perp \text{ or Output} = \mathcal{U}(P)] \geq 1 - \varepsilon.$$

Security breach



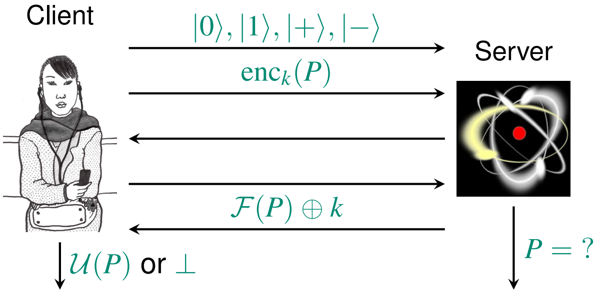
- We run the protocol of [BFK 2009].
- We restrict the programs to efficiently verifiable ones, e.g., factoring, finding a witness for a positive NP instance.
- This satisfies our security criteria.

Security breach



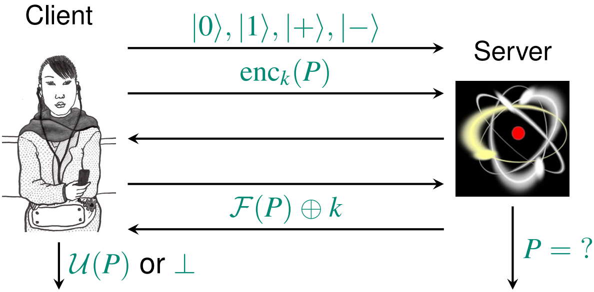
- We run the protocol of [BFK 2009].
- We restrict the programs to efficiently verifiable ones, e.g., factoring, finding a witness for a positive NP instance.
- This satisfies our security criteria.

Security breach



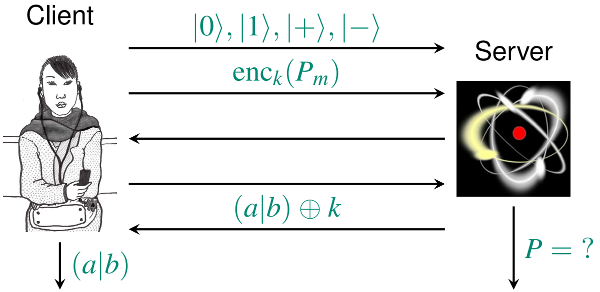
- We run the protocol of [BFK 2009].
- We restrict the programs to efficiently verifiable ones, e.g., factoring, finding a witness for a positive NP instance.
- This satisfies our security criteria.

Security breach



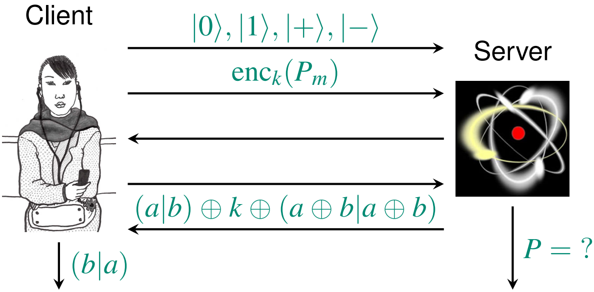
- We run the protocol of [BFK 2009].
- We restrict the programs to efficiently verifiable ones, e.g., factoring, finding a witness for a positive NP instance.
- This satisfies our security criteria.

Security breach



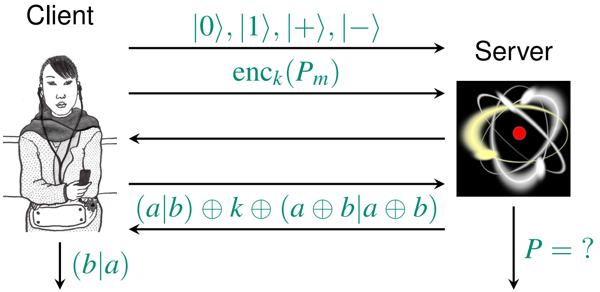
- The client wants to factor either $m = ab$ or $n = pq$.
- The server XORs $(a \oplus b | a \oplus b)$ to the final message.
- If the input was m, the client accepts $(b|a)$.
- If the input was n, the client rejects.
- If the server learns if the client accepts, the server learns the input!

Security breach



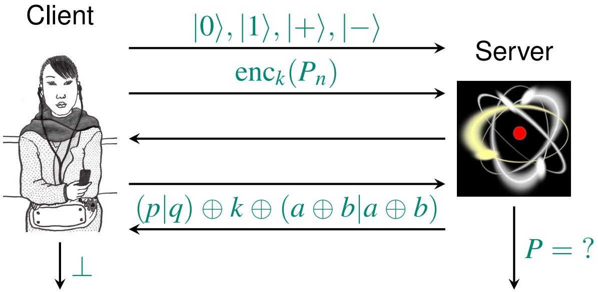
- The client wants to factor either $m = ab$ or $n = pq$.
- The server XORs $(a \oplus b|a \oplus b)$ to the final message.
- If the input was m, the client accepts $(b|a)$.
- If the input was n, the client rejects.
- If the server learns if the client accepts, the server learns the input!

Security breach



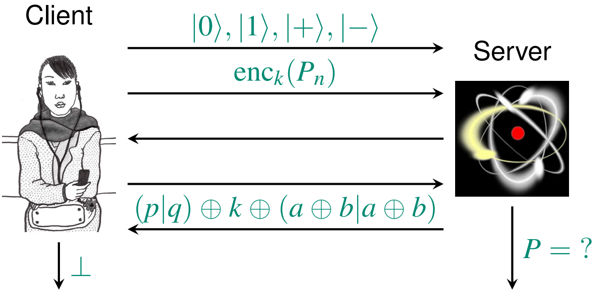
- The client wants to factor either $m = ab$ or $n = pq$.
- The server XORs $(a \oplus b|a \oplus b)$ to the final message.
- If the input was m, the client accepts $(b|a)$.
- If the input was n, the client rejects.
- If the server learns if the client accepts, the server learns the input!

Security breach



- The client wants to factor either $m = ab$ or $n = pq$.
- The server XORs $(a \oplus b|a \oplus b)$ to the final message.
- If the input was m, the client accepts $(b|a)$.
- If the input was n, the client rejects.
- If the server learns if the client accepts, the server learns the input!

Security breach



- The client wants to factor either $m = ab$ or $n = pq$.
- The server XORs $(a \oplus b|a \oplus b)$ to the final message.
- If the input was m, the client accepts $(b|a)$.
- If the input was n, the client rejects.
- If the server learns if the client accepts, the server learns the input!

Authenticate-then-encrypt

Similar security breach for authenticate-then-encrypt [Bellare, Namprempre 2000; Krawczyk 2001]: one can construct protocols such that,

- the adversary learns nothing about the message from the cipher, $H(M|C) = H(M)$,
- with high probability the receiver does not accept modified messages, $\Pr[m_B = m_A$ or $m_B = \perp] \geq 1 - \varepsilon$,
- if the adversary learns if the (modified) ciphertext was successfully authenticated, he learns a bit of the message.

Composable security

Abstract Cryptography (AC) [Maurer, Renner 2011]

- Views cryptography as a resource theory: a protocol π constructs a (strong) resource $\mathcal{S}$ from a (weak) resource $\mathcal{R}$.

$$\mathcal{R} \xrightarrow{\pi,\varepsilon} \mathcal{S}.$$

- Resources are abstract systems that can be instantiated as desired (e.g., classical or quantum computation, synchronous or asynchronous communication).

Theorem (Sequential and parallel composition)

- $\mathcal{R} \xrightarrow{\pi,\varepsilon} \mathcal{S}$ *and* $\mathcal{S} \xrightarrow{\pi',\varepsilon'} \mathcal{T} \implies \mathcal{R} \xrightarrow{\pi' \circ \pi,\varepsilon+\varepsilon'} \mathcal{T}.$
- $\mathcal{R} \xrightarrow{\pi,\varepsilon} \mathcal{S}$ *and* $\mathcal{R}' \xrightarrow{\pi',\varepsilon'} \mathcal{S} \implies \mathcal{R}\|\mathcal{R}' \xrightarrow{\pi\|\pi',\varepsilon+\varepsilon'} \mathcal{S}\|\mathcal{S}'.$

Composable security

Abstract Cryptography (AC) [Maurer, Renner 2011]

- Views cryptography as a resource theory: a protocol π constructs a (strong) resource $\mathcal{S}$ from a (weak) resource $\mathcal{R}$.

$$\mathcal{R} \xrightarrow{\pi,\varepsilon} \mathcal{S}.$$

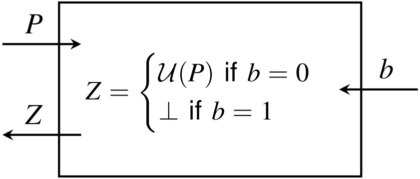
- Resources are abstract systems that can be instantiated as desired (e.g., classical or quantum computation, synchronous or asynchronous communication).

Theorem (Sequential and parallel composition)

- $\mathcal{R} \xrightarrow{\pi,\varepsilon} \mathcal{S}$ *and* $\mathcal{S} \xrightarrow{\pi',\varepsilon'} \mathcal{T} \implies \mathcal{R} \xrightarrow{\pi'\circ\pi,\varepsilon+\varepsilon'} \mathcal{T}$.
- $\mathcal{R} \xrightarrow{\pi,\varepsilon} \mathcal{S}$ *and* $\mathcal{R}' \xrightarrow{\pi',\varepsilon'} \mathcal{S} \implies \mathcal{R}\|\mathcal{R}' \xrightarrow{\pi\|\pi',\varepsilon+\varepsilon'} \mathcal{S}\|\mathcal{S}'$.

Ideal DQC resource: blindness + verifiability

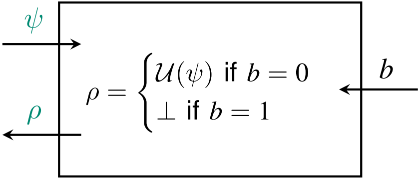
Ideal resource $\mathcal{S}^{\text{blind}}_{\text{verif}}$



- The client inputs the program P.
- The (dishonest) server decides if the client gets the correct outcome or an error message $\perp$.
- The ideal resource provides the output.
- This also works with quantum inputs and outputs.
- An honest server is modeled by a filter $\sharp$ setting $b = 0$.

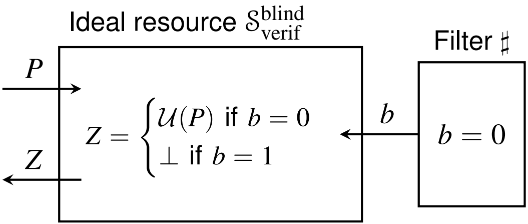
Ideal DQC resource: blindness + verifiability

Ideal resource $\mathcal{S}^{\text{blind}}_{\text{verif}}$



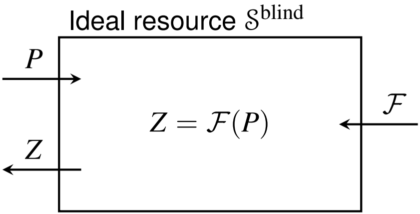
- The client inputs the program P.
- The (dishonest) server decides if the client gets the correct outcome or an error message $\perp$.
- The ideal resource provides the output.
- This also works with quantum inputs and outputs.
- An honest server is modeled by a filter $\sharp$ setting $b = 0$.

Ideal DQC resource: blindness + verifiability



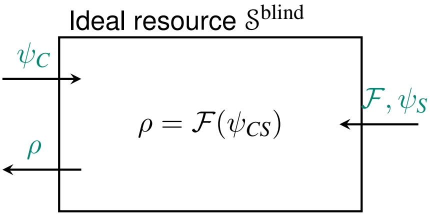
- The client inputs the program P.
- The (dishonest) server decides if the client gets the correct outcome or an error message $\perp$.
- The ideal resource provides the output.
- This also works with quantum inputs and outputs.
- An honest server is modeled by a filter $\sharp$ setting $b = 0$.

Ideal DQC resource: blindness only



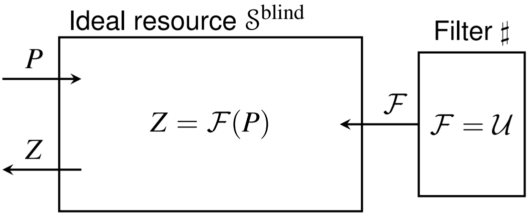
- The client inputs the program P.
- The (dishonest) server decides what computation is effectively run.
- The ideal resource provides the output.
- This also works with quantum inputs and outputs.
- An honest server is modeled by a filter $\sharp$ setting $\mathcal{F} = \mathcal{U}$.

Ideal DQC resource: blindness only



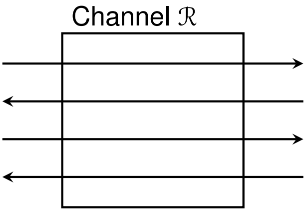
- The client inputs the program P.
- The (dishonest) server decides what computation is effectively run.
- The ideal resource provides the output.
- This also works with quantum inputs and outputs.
- An honest server is modeled by a filter $\sharp$ setting $\mathcal{F} = \mathcal{U}$.

Ideal DQC resource: blindness only



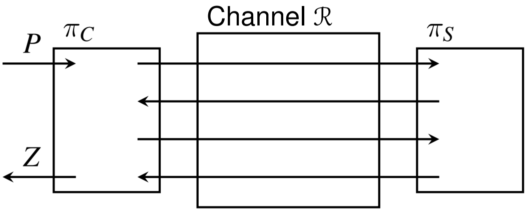
- The client inputs the program P.
- The (dishonest) server decides what computation is effectively run.
- The ideal resource provides the output.
- This also works with quantum inputs and outputs.
- An honest server is modeled by a filter $\sharp$ setting $\mathcal{F} = \mathcal{U}$.

Real world



- The only resource available is communication channels $\mathcal{R}$.
- The client and server run the joint protocol (π_C, π_S).

Real world



- The only resource available is communication channels $\mathcal{R}$.
- The client and server run the joint protocol (π_C, π_S).

Security definition

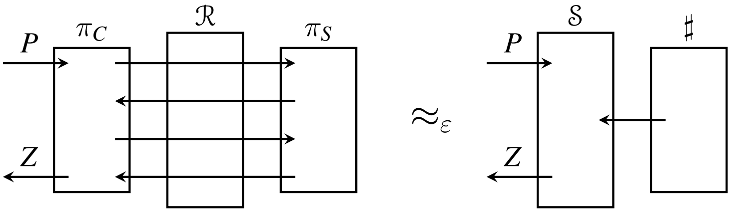
Definition

A protocol $\pi = (\pi_C, \pi_S)$ constructs $\mathcal{S}$ from $\mathcal{R}$ within ε, $\mathcal{R} \xrightarrow{\pi,\varepsilon} \mathcal{S}$, if

$$\pi_C \mathcal{R} \pi_S \approx_\varepsilon \mathcal{S}\sharp, \qquad \text{(correctness)}$$

and if there exists a simulator σ such that

$$\pi_C \mathcal{R} \approx_\varepsilon \mathcal{S}\sigma. \qquad \text{(security)}$$



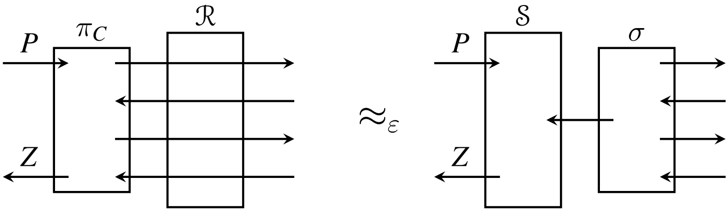
Definition

A protocol $\pi = (\pi_C, \pi_S)$ constructs $\mathcal{S}$ from $\mathcal{R}$ within ε, $\mathcal{R} \xrightarrow{\pi,\varepsilon} \mathcal{S}$, if

$$\pi_C \mathcal{R} \pi_S \approx_\varepsilon \mathcal{S}\sharp, \qquad \text{(correctness)}$$

and if there exists a simulator σ such that

$$\pi_C \mathcal{R} \approx_\varepsilon \mathcal{S}\sigma. \qquad \text{(security)}$$



Security definition

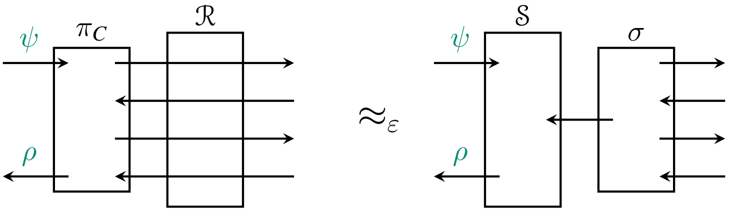
Definition

A protocol $\pi = (\pi_C, \pi_S)$ constructs $\mathcal{S}$ from $\mathcal{R}$ within ε, $\mathcal{R} \xrightarrow{\pi,\varepsilon} \mathcal{S}$, if

$$\pi_C \mathcal{R} \pi_S \approx_\varepsilon \mathcal{S}\sharp, \qquad \text{(correctness)}$$

and if there exists a simulator σ such that

$$\pi_C \mathcal{R} \approx_\varepsilon \mathcal{S}\sigma. \qquad \text{(security)}$$

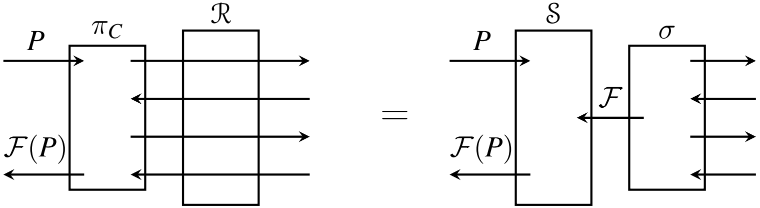


New (composable) security proofs

In the case of DQC protocols that provide blindness only

Theorem

- *The protocol of [Broadbent, Fitzsimons, Kashefi 2009] is perfectly blind.*
- *The protocol of [Morimae, Fujii 2013] is perfectly blind.*



Strengthening the ad hoc security definitions

Blindness (informal)

The server S obtains (approximately) no information about the program P,

$$H(P|S) \approx_\varepsilon H(P).$$

Verifiability (informal)

(With high probability) the client does not accept a wrong result,

$$\Pr[\text{Output} = \perp \text{ or Output} = \mathcal{U}(P)] \geq 1 - \varepsilon.$$

Independent verifiability (informal)

(With high probability) the server can guess if the client will accept or reject the result,

$$\Pr[\text{server guess} = \text{client decision}] \geq 1 - \varepsilon.$$

Strengthening the ad hoc security definitions

Blindness (informal)

The server S obtains (approximately) no information about the program P,

$$H(P|S) \approx_{\varepsilon} H(P).$$

Verifiability (informal)

(With high probability) the client does not accept a wrong result,

$$\Pr[\text{Output} = \perp \text{ or Output} = \mathcal{U}(P)] \geq 1 - \varepsilon.$$

Independent verifiability (informal)

(With high probability) the server can guess if the client will accept or reject the result,

$$\Pr[\text{server guess} = \text{client decision}] \geq 1 - \varepsilon.$$

Security reduction

Theorem

If a DQC protocol is ε_1-blind and ε_2-independent ε_3-verifiable for all inputs ψ_{CQ}, where C is classical and Q is quantum, then it is δN^2-secure, where $\delta = 2\varepsilon_1 + 2\varepsilon_2 + 4\sqrt{2\varepsilon_3}$ and $N = \dim Q$.