Public-key Cryptography
Theory and Practice

Abhijit Das

Department of Computer Science and Engineering
Indian Institute of Technology Kharagpur

Chapter 8: Quantum Computation and Cryptography
What is Quantum Cryptography
What is Quantum Cryptography

- Based on the paradigm of quantum computation.
What is Quantum Cryptography

- Based on the paradigm of quantum computation.
- Governed by the laws of quantum mechanics.
What is Quantum Cryptography

- Based on the paradigm of quantum computation.
- Governed by the laws of quantum mechanics.

**Quantum cryptanalysis:** Probabilistic polynomial-time algorithms are known to solve the integer factorization and finite field discrete logarithm problems.
What is Quantum Cryptography

- Based on the paradigm of quantum computation.
- Governed by the laws of quantum mechanics.

**Quantum cryptanalysis**: Probabilistic polynomial-time algorithms are known to solve the integer factorization and finite field discrete logarithm problems.

**Quantum cryptography**: A provably secure key exchange method is based upon quantum computation.
What is Quantum Cryptography

- Based on the paradigm of quantum computation.
- Governed by the laws of quantum mechanics.

Quantum cryptanalysis: Probabilistic polynomial-time algorithms are known to solve the integer factorization and finite field discrete logarithm problems.

Quantum cryptography: A provably secure key exchange method is based upon quantum computation.

It is not known how to build a quantum computer.
What is Quantum Cryptography

- Based on the paradigm of quantum computation.
- Governed by the laws of quantum mechanics.

**Quantum cryptanalysis:** Probabilistic polynomial-time algorithms are known to solve the integer factorization and finite field discrete logarithm problems.

**Quantum cryptography:** A provably secure key exchange method is based upon quantum computation.

- It is not known how to build a quantum computer.
- Some partial implementations are known.
There was a time when the newspapers said that only twelve men understood the theory of relativity. I do not believe there ever was such a time . . . On the other hand, I think I can safely say that nobody understands quantum mechanics.

— Richard Feynman
(The Character of Physical Law, BBC, 1965)
Quantum-mechanical Systems
Quantum-mechanical Systems

- A **system** is specified by a finite-dimensional normalized vector of complex numbers:
A **system** is specified by a finite-dimensional normalized vector of complex numbers:

\[(z_0, z_1, \ldots, z_{n-1}) \text{ with } z_i \in \mathbb{C} \text{ and } \sum_{i=0}^{n-1} |z_i|^2 = 1.\]
A **system** is specified by a finite-dimensional normalized vector of complex numbers:

\[(z_0, z_1, \ldots, z_{n-1}) \text{ with } z_i \in \mathbb{C} \text{ and } \sum_{i=0}^{n-1} |z_i|^2 = 1.\]

Choose an orthonormal basis \(B\) of \(\mathbb{C}^n\). Denote the elements of \(B\) as \(|0\rangle, |1\rangle, \ldots, |n-1\rangle\). For example,
A system is specified by a finite-dimensional normalized vector of complex numbers:

\[(z_0, z_1, \ldots, z_{n-1}) \text{ with } z_i \in \mathbb{C} \text{ and } \sum_{i=0}^{n-1} |z_i|^2 = 1.\]

Choose an orthonormal basis \(B\) of \(\mathbb{C}^n\). Denote the elements of \(B\) as \(|0\rangle, |1\rangle, \ldots, |n-1\rangle\). For example,

\[
|0\rangle = (1, 0, 0, \ldots, 0), \\
|1\rangle = (0, 1, 0, \ldots, 0), \\
|2\rangle = (0, 0, 1, \ldots, 0), \\
\ldots \\
|n-1\rangle = (0, 0, 0, \ldots, 1).
\]
A *system* is specified by a finite-dimensional normalized vector of complex numbers:

\[(z_0, z_1, \ldots, z_{n-1}) \text{ with } z_i \in \mathbb{C} \text{ and } \sum_{i=0}^{n-1} |z_i|^2 = 1.\]

Choose an orthonormal basis \(B\) of \(\mathbb{C}^n\). Denote the elements of \(B\) as \(|0\rangle, |1\rangle, \ldots, |n-1\rangle\). For example,

\[
|0\rangle = (1, 0, 0, \ldots, 0),
|1\rangle = (0, 1, 0, \ldots, 0),
|2\rangle = (0, 0, 1, \ldots, 0),
\]

\[
\ldots
\]

\[
|n-1\rangle = (0, 0, 0, \ldots, 1).
\]

The state of a system is \(z_0|0\rangle + z_1|1\rangle + \cdots + z_{n-1}|n-1\rangle\) with \(z_i \in \mathbb{C}\) and \(\sum_{i=0}^{n-1} |z_i|^2 = 1\).
Quantum Bit (qubit)

Quantum Cryptography
Quantum Cryptanalysis
Quantum Bits and Registers

Abhijit Das
A classical bit (cbit) can take two values: 0 and 1.
Quantum Bit (qubit)

- A **classical bit** (cbit) can take two values: 0 and 1.
- A **quantum bit** (qubit) is a normalized 2-dimensional vector of complex numbers.
Quantum Bit (qubit)

- A **classical bit** (cbit) can take two values: 0 and 1.
- A **quantum bit** (qubit) is a normalized 2-dimensional vector of complex numbers.
- The basis states are $|0\rangle$ and $|1\rangle$. 
Quantum Bit (qubit)

- A **classical bit (cbit)** can take two values: 0 and 1.
- A **quantum bit (qubit)** is a normalized 2-dimensional vector of complex numbers.
- The basis states are $|0\rangle$ and $|1\rangle$.
- All the values that a qubit may have are $a|0\rangle + b|1\rangle$ with $a^2 + b^2 = 1$. 
Quantum Bit (qubit)

- A classical bit (cbit) can take two values: 0 and 1.
- A quantum bit (qubit) is a normalized 2-dimensional vector of complex numbers.
- The basis states are $|0\rangle$ and $|1\rangle$.
- All the values that a qubit may have are $a|0\rangle + b|1\rangle$ with $a^2 + b^2 = 1$.
- Possible realizations:
Quantum Bit (qubit)

- A **classical bit** (cbit) can take two values: 0 and 1.
- A **quantum bit** (qubit) is a normalized 2-dimensional vector of complex numbers.
- The basis states are $|0\rangle$ and $|1\rangle$.
- All the values that a qubit may have are $a|0\rangle + b|1\rangle$ with $a^2 + b^2 = 1$.
- Possible realizations:
  - Spin of an electron ($|\text{Up}\rangle$ and $|\text{Down}\rangle$)
A **classical bit** (cbit) can take two values: 0 and 1.

A **quantum bit** (qubit) is a normalized 2-dimensional vector of complex numbers.

The basis states are $|0\rangle$ and $|1\rangle$.

All the values that a qubit may have are $a|0\rangle + b|1\rangle$ with $a^2 + b^2 = 1$.

Possible realizations:
- Spin of an electron ($|\text{Up}\rangle$ and $|\text{Down}\rangle$)
- Polarization of a photon
Quantum Bit (qubit)

- A **classical bit** (cbit) can take two values: 0 and 1.
- A **quantum bit** (qubit) is a normalized 2-dimensional vector of complex numbers.
- The basis states are $|0\rangle$ and $|1\rangle$.
- All the values that a qubit may have are $a|0\rangle + b|1\rangle$ with $a^2 + b^2 = 1$.
- Possible realizations:
  - Spin of an electron ($|Up\rangle$ and $|Down\rangle$)
  - Polarization of a photon
- Conceptual example:
Quantum Bit (qubit)

- A classical bit (cbit) can take two values: 0 and 1.
- A quantum bit (qubit) is a normalized 2-dimensional vector of complex numbers.
- The basis states are $|0\rangle$ and $|1\rangle$.
- All the values that a qubit may have are $a|0\rangle + b|1\rangle$ with $a^2 + b^2 = 1$.
- Possible realizations:
  - Spin of an electron ($|\text{Up}\rangle$ and $|\text{Down}\rangle$)
  - Polarization of a photon
- Conceptual example:
  - Schrödinger cat ($|\text{Alive}\rangle$ and $|\text{Dead}\rangle$)
Quantum Bit (qubit)

- A **classical bit** (**cbit**) can take two values: 0 and 1.
- A **quantum bit** (**qubit**) is a normalized 2-dimensional vector of complex numbers.
- The basis states are $|0\rangle$ and $|1\rangle$.
- All the values that a qubit may have are $a|0\rangle + b|1\rangle$ with $a^2 + b^2 = 1$.
- Possible realizations:
  - Spin of an electron ($|\text{Up}\rangle$ and $|\text{Down}\rangle$)
  - Polarization of a photon
- Conceptual example:
  - **Schrödinger cat** ($|\text{Alive}\rangle$ and $|\text{Dead}\rangle$)
  - The cat may be in the state $(|\text{Alive}\rangle + |\text{Dead}\rangle)/\sqrt{2}$. 
Composite Systems
A is a system with basis $|0\rangle_A, |1\rangle_A, \ldots, |m - 1\rangle_A$. 
Composite Systems

- $A$ is a system with basis $|0\rangle_A, |1\rangle_A, \ldots, |m - 1\rangle_A$.
- $B$ is a system with basis $|0\rangle_B, |1\rangle_B, \ldots, |n - 1\rangle_B$. 

Public-key Cryptography: Theory and Practice
Abhijit Das
A is a system with basis $|0\rangle_A, |1\rangle_A, \ldots, |m-1\rangle_A$.

B is a system with basis $|0\rangle_B, |1\rangle_B, \ldots, |n-1\rangle_B$.

AB is a system with two parts A and B.
Composite Systems

- $A$ is a system with basis $|0\rangle_A, |1\rangle_A, \ldots, |m-1\rangle_A$.
- $B$ is a system with basis $|0\rangle_B, |1\rangle_B, \ldots, |n-1\rangle_B$.
- $AB$ is a system with two parts $A$ and $B$.
- $AB$ is an $mn$-dimensional system with basis
  \[ |i\rangle_A \otimes |j\rangle_B = |i\rangle_A |j\rangle_B = |ij\rangle_{AB} = |ij\rangle. \]
Composite Systems

- $A$ is a system with basis $|0\rangle_A, |1\rangle_A, \ldots, |m-1\rangle_A$.
- $B$ is a system with basis $|0\rangle_B, |1\rangle_B, \ldots, |n-1\rangle_B$.
- $AB$ is a system with two parts $A$ and $B$.
- $AB$ is an $mn$-dimensional system with basis $|i\rangle_A \otimes |j\rangle_B = |i\rangle_A |j\rangle_B = |ij\rangle_{AB} = |ij\rangle$.
- State of $AB$: $\sum_{i,j} a_{ij} |ij\rangle$ with $\sum_{i,j} |a_{ij}|^2 = 1$. 
Composite Systems

- $A$ is a system with basis $|0\rangle_A, |1\rangle_A, \ldots, |m-1\rangle_A$.
- $B$ is a system with basis $|0\rangle_B, |1\rangle_B, \ldots, |n-1\rangle_B$.
- $AB$ is a system with two parts $A$ and $B$.
- $AB$ is an $mn$-dimensional system with basis
  \[ |i\rangle_A \otimes |j\rangle_B = |i\rangle_A|j\rangle_B = |ij\rangle_{AB} = |ij\rangle. \]
- State of $AB$: $\sum_{i,j} a_{ij} |ij\rangle$ with $\sum_{i,j} |a_{ij}|^2 = 1$.

Let $A_1, A_2, \ldots, A_k$ be systems of dimensions $n_1, n_2, \ldots, n_k$. 

### Quantum Cryptography

### Laws of Quantum Mechanics

### Quantum Bits and Registers

### Operations on a System

### Measurement of a System

### Composite Systems

- $A$ is a system with basis $|0\rangle_A, |1\rangle_A, \ldots, |m-1\rangle_A$.
- $B$ is a system with basis $|0\rangle_B, |1\rangle_B, \ldots, |n-1\rangle_B$.
- $AB$ is a system with two parts $A$ and $B$.
- $AB$ is an $mn$-dimensional system with basis
  \[ |i\rangle_A \otimes |j\rangle_B = |i\rangle_A|j\rangle_B = |ij\rangle_{AB} = |ij\rangle. \]
- State of $AB$: $\sum_{i,j} a_{ij} |ij\rangle$ with $\sum_{i,j} |a_{ij}|^2 = 1$.

Let $A_1, A_2, \ldots, A_k$ be systems of dimensions $n_1, n_2, \ldots, n_k$. 

### Public-key Cryptography: Theory and Practice

### Abhijit Das
### Composite Systems

- **A** is a system with basis $|0\rangle_A, |1\rangle_A, \ldots, |m-1\rangle_A$.
- **B** is a system with basis $|0\rangle_B, |1\rangle_B, \ldots, |n-1\rangle_B$.
- **AB** is a system with two parts **A** and **B**.
- **AB** is an $mn$-dimensional system with basis $|i\rangle_A \otimes |j\rangle_B = |i\rangle_A |j\rangle_B = |ij\rangle_{AB} = |ij\rangle$.
- State of **AB**: $\sum_{i,j} a_{ij} |ij\rangle$ with $\sum_{i,j} |a_{ij}|^2 = 1$.

Let $A_1, A_2, \ldots, A_k$ be systems of dimensions $n_1, n_2, \ldots, n_k$.

- **A_1A_2\ldots A_k** is the $n_1 n_2 \cdots n_k$-dimensional system with basis $|j_1\rangle_1 \otimes |j_2\rangle_2 \otimes \cdots \otimes |j_k\rangle_k = |j_1\rangle_1 |j_2\rangle_2 \cdots |j_k\rangle_k = |j_1 j_2 \cdots j_k\rangle$. 
Quantum Registers
An $n$-bit quantum register $R$ has exactly $n$ qubits.
Quantum Registers

- An $n$-bit quantum register $R$ has exactly $n$ qubits.
- $R$ is a normalized $2^n$-dimensional vector.
An $n$-bit quantum register $R$ has exactly $n$ qubits.

$R$ is a normalized $2^n$-dimensional vector.

The basis states are

$$|j_1\rangle \otimes |j_2\rangle \otimes \cdots \otimes |j_n\rangle = |j_1j_2\cdots j_n\rangle.$$
Quantum Registers

- An $n$-bit quantum register $R$ has exactly $n$ qubits.
- $R$ is a normalized $2^n$-dimensional vector.
- The basis states are
  $$|j_1\rangle \otimes |j_2\rangle \otimes \cdots \otimes |j_n\rangle = |j_1\rangle|j_2\rangle \cdots |j_n\rangle = |j_1j_2 \cdots j_n\rangle.$$  
- The basis states may be renamed as $|0\rangle, |1\rangle, \ldots, |2^n - 1\rangle$. 
Quantum Registers

- An $n$-bit quantum register $R$ has exactly $n$ qubits.
- $R$ is a normalized $2^n$-dimensional vector.
- The basis states are
  $$|j_1\rangle \otimes |j_2\rangle \otimes \cdots \otimes |j_n\rangle = |j_1j_2\cdots j_n\rangle.$$  
- The basis states may be renamed as $|0\rangle, |1\rangle, \ldots, |2^n - 1\rangle$.
- The basis states correspond to the classical values of an $n$-bit register.
Quantum Registers

- An $n$-bit quantum register $R$ has exactly $n$ qubits.
- $R$ is a normalized $2^n$-dimensional vector.
- The basis states are
  $$|j_1\rangle \otimes |j_2\rangle \otimes \cdots \otimes |j_n\rangle = |j_1j_2\ldots j_n\rangle.$$  
- The basis states may be renamed as $|0\rangle, |1\rangle, \ldots, |2^n - 1\rangle$.
- The basis states correspond to the classical values of an $n$-bit register.
- A general state for $R$ is
  $$|\psi\rangle = \sum_{i=0}^{2^n-1} a_i |i\rangle \text{ with } a_i \in \mathbb{C} \text{ and } \sum_{i=0}^{2^n-1} |a_i|^2 = 1.$$
Entanglement
Entanglement

Let $R = AB$ be a 2-bit quantum register.
Entanglement

- Let $R = AB$ be a 2-bit quantum register.
- A general state for $R$ is $c_0|0\rangle + c_1|1\rangle + c_2|2\rangle + c_3|3\rangle$. 
Entanglement

- Let $R = AB$ be a 2-bit quantum register.
- A general state for $R$ is $c_0|0\rangle + c_1|1\rangle + c_2|2\rangle + c_3|3\rangle$.
- This can be written in the form

\[
(a_0|0\rangle + a_1|1\rangle)(b_0|0\rangle + b_1|1\rangle)
= a_0 b_0 |0\rangle + a_0 b_1 |1\rangle + a_1 b_0 |2\rangle + a_1 b_1 |3\rangle
\]

if and only if $c_0 c_3 = c_1 c_2$. 
Entanglement

- Let $R = AB$ be a 2-bit quantum register.
- A general state for $R$ is $c_0 |0\rangle + c_1 |1\rangle + c_2 |2\rangle + c_3 |3\rangle$.
- This can be written in the form
  \[
  (a_0 |0\rangle + a_1 |1\rangle)(b_0 |0\rangle + b_1 |1\rangle)
  = a_0 b_0 |0\rangle + a_0 b_1 |1\rangle + a_1 b_0 |2\rangle + a_1 b_1 |3\rangle
  \]
  if and only if $c_0 c_3 = c_1 c_2$.
- If $c_0 c_3 \neq c_1 c_2$, then the bits $A$ and $B$ do not possess individual states.
Entanglement

- Let $R = AB$ be a 2-bit quantum register.
- A general state for $R$ is $c_0|0\rangle + c_1|1\rangle + c_2|2\rangle + c_3|3\rangle$.
- This can be written in the form

$$
(a_0|0\rangle + a_1|1\rangle)(b_0|0\rangle + b_1|1\rangle)
= a_0 b_0|0\rangle + a_0 b_1|1\rangle + a_1 b_0|2\rangle + a_1 b_1|3\rangle
$$

if and only if $c_0 c_3 = c_1 c_2$.

- If $c_0 c_3 \neq c_1 c_2$, then the bits $A$ and $B$ do not possess individual states.

- An $n$-bit quantum register is called **entangled** if no set of fewer than its $n$ qubits possesses an individual state.
Entanglement

- Let $R = AB$ be a 2-bit quantum register.
- A general state for $R$ is $c_0 |0\rangle + c_1 |1\rangle + c_2 |2\rangle + c_3 |3\rangle$.
- This can be written in the form
  \[
  (a_0 |0\rangle + a_1 |1\rangle)(b_0 |0\rangle + b_1 |1\rangle)
  = a_0 b_0 |0\rangle + a_0 b_1 |1\rangle + a_1 b_0 |2\rangle + a_1 b_1 |3\rangle
  \]
  if and only if $c_0 c_3 = c_1 c_2$.
- If $c_0 c_3 \neq c_1 c_2$, then the bits $A$ and $B$ do not possess individual states.
- An $n$-bit quantum register is called entangled if no set of fewer than its $n$ qubits possesses an individual state.
- Entanglement with surroundings poses the biggest challenge for realizing quantum computers.
Evolution of a System
Evolution of a System

- The conjugate transpose of a square matrix $U = (u_{ij})$ with complex entries is denoted by $U^\dagger = (\overline{u_{ji}})$. 
The conjugate transpose of a square matrix $U = (u_{ij})$ with complex entries is denoted by $U^\dagger = (\overline{u_{ji}})$.

$U$ is called **unitary** if $UU^\dagger = U^\dagger U = I$. 

**Evolution of a System**
The conjugate transpose of a square matrix $U = (u_{ij})$ with complex entries is denoted by $U^\dagger = (\overline{u_{ji}})$.

$U$ is called **unitary** if $UU^\dagger = U^\dagger U = I$.

Every unitary matrix $U$ is invertible with $U^{-1} = U^\dagger$. 

The conjugate transpose of a square matrix $U = (u_{ij})$ with complex entries is denoted by $U^\dagger = (\overline{u_{ji}})$.

$U$ is called **unitary** if $UU^\dagger = U^\dagger U = I$.

Every unitary matrix $U$ is invertible with $U^{-1} = U^\dagger$.

Any operation on a quantum-mechanical system is unitary.
The conjugate transpose of a square matrix $U = (u_{ij})$ with complex entries is denoted by $U^\dagger = (\overline{u_{ji}})$. 

$U$ is called unitary if $UU^\dagger = U^\dagger U = I$. 

Every unitary matrix $U$ is invertible with $U^{-1} = U^\dagger$. 

Any operation on a quantum-mechanical system is unitary. 

In particular, all operations on a quantum-mechanical system are invertible.
The conjugate transpose of a square matrix $U = (u_{ij})$ with complex entries is denoted by $U^\dagger = (\overline{u_{ji}})$.

$U$ is called **unitary** if $UU^\dagger = U^\dagger U = I$.

Every unitary matrix $U$ is invertible with $U^{-1} = U^\dagger$.

Any operation on a quantum-mechanical system is unitary.

In particular, all operations on a quantum-mechanical system are invertible.

**No-cloning theorem:** It is impossible to copy the contents of a quantum register to another. (The transformation $|\psi\rangle|\varphi\rangle \mapsto |\psi\rangle|\psi\rangle$ is not invertible.)
Examples of Unitary Operators on a Qubit

<table>
<thead>
<tr>
<th>Operator</th>
<th>Transformation</th>
<th>Matrix</th>
</tr>
</thead>
</table>

Public-key Cryptography: Theory and Practice  Abhijit Das
### Examples of Unitary Operators on a Qubit

<table>
<thead>
<tr>
<th>Operator</th>
<th>Transformation</th>
<th>Matrix</th>
</tr>
</thead>
<tbody>
<tr>
<td>Identity</td>
<td>$I</td>
<td>0\rangle =</td>
</tr>
</tbody>
</table>
### Examples of Unitary Operators on a Qubit

<table>
<thead>
<tr>
<th>Operator</th>
<th>Transformation</th>
<th>Matrix</th>
</tr>
</thead>
<tbody>
<tr>
<td>Identity</td>
<td>$I</td>
<td>0\rangle =</td>
</tr>
<tr>
<td>Exchange</td>
<td>$I</td>
<td>0\rangle =</td>
</tr>
</tbody>
</table>
### Examples of Unitary Operators on a Qubit

<table>
<thead>
<tr>
<th>Operator</th>
<th>Transformation</th>
<th>Matrix</th>
</tr>
</thead>
</table>
| **Identity** | \( I|0\rangle = |0\rangle, \; I|1\rangle = |1\rangle \) | \[
\begin{pmatrix}
1 & 0 \\
0 & 1
\end{pmatrix}
\] |
| **Exchange** | \( I|0\rangle = |1\rangle, \; I|1\rangle = |0\rangle \) | \[
\begin{pmatrix}
0 & 1 \\
1 & 0
\end{pmatrix}
\] |
| **Z** | \( Z|0\rangle = |0\rangle, \; Z|1\rangle = -|1\rangle \) | \[
\begin{pmatrix}
1 & 0 \\
0 & -1
\end{pmatrix}
\] |
### Examples of Unitary Operators on a Qubit

<table>
<thead>
<tr>
<th>Operator</th>
<th>Transformation</th>
<th>Matrix</th>
</tr>
</thead>
</table>
| Identity   | $I|0\rangle = |0\rangle$, $I|1\rangle = |1\rangle$ | \[
\begin{pmatrix}
1 & 0 \\
0 & 1
\end{pmatrix}
\] |
| Exchange   | $I|0\rangle = |1\rangle$, $I|1\rangle = |0\rangle$ | \[
\begin{pmatrix}
0 & 1 \\
1 & 0
\end{pmatrix}
\] |
| $Z$        | $Z|0\rangle = |0\rangle$, $Z|1\rangle = -|1\rangle$ | \[
\begin{pmatrix}
1 & 0 \\
0 & -1
\end{pmatrix}
\] |
| Hadamard   | $H|0\rangle = \frac{1}{\sqrt{2}}(|0\rangle + |1\rangle)$, $H|1\rangle = \frac{1}{\sqrt{2}}(|0\rangle - |1\rangle)$ | \[
\begin{pmatrix}
1 & 1 \\
1 & -1
\end{pmatrix}
\] |
### Examples of Unitary Operators on a Qubit

<table>
<thead>
<tr>
<th>Operator</th>
<th>Transformation</th>
<th>Matrix</th>
</tr>
</thead>
</table>
| Identity        | $I|0\rangle = |0\rangle$, $I|1\rangle = |1\rangle$ | \[
\begin{pmatrix}
1 & 0 \\
0 & 1
\end{pmatrix}
\] |
| Exchange        | $I|0\rangle = |1\rangle$, $I|1\rangle = |0\rangle$ | \[
\begin{pmatrix}
0 & 1 \\
1 & 0
\end{pmatrix}
\] |
| $Z$             | $Z|0\rangle = |0\rangle$, $Z|1\rangle = -|1\rangle$ | \[
\begin{pmatrix}
1 & 0 \\
0 & -1
\end{pmatrix}
\] |
| Hadamard        | $H|0\rangle = \frac{1}{\sqrt{2}}(|0\rangle + |1\rangle)$ | $\frac{1}{\sqrt{2}} \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix}$ |
|                 | $H|1\rangle = \frac{1}{\sqrt{2}}(|0\rangle - |1\rangle)$ |               |
| $\sqrt{X}$      | $\sqrt{X}|0\rangle = \frac{1}{1+i}(|0\rangle + i|1\rangle)$ | $\frac{1}{1+i} \begin{pmatrix} 1 & i \\ i & 1 \end{pmatrix}$ |
|                 | $\sqrt{X}|1\rangle = \frac{1}{1+i}(i|0\rangle + |1\rangle)$ |               |
Examples of Unitary Operators (contd)
Examples of Unitary Operators (contd)

Let $|\psi\rangle = a|0\rangle + b|1\rangle$ be a state of a qubit.
Examples of Unitary Operators (contd)

Let $|\psi\rangle = a|0\rangle + b|1\rangle$ be a state of a qubit.

\[
H|\psi\rangle = H(a|0\rangle + b|1\rangle) = aH|0\rangle + bH|1\rangle = a \left[ \frac{1}{\sqrt{2}} (|0\rangle + |1\rangle) \right] + b \left[ \frac{1}{\sqrt{2}} (|0\rangle - |1\rangle) \right] = \left( \frac{a + b}{\sqrt{2}} \right) |0\rangle + \left( \frac{a - b}{\sqrt{2}} \right) |1\rangle = (a \ b) \frac{1}{\sqrt{2}} \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix} \begin{pmatrix} |0\rangle \\ |1\rangle \end{pmatrix}
\]
Measurement

The Born Rule
The Born Rule

Let $A$ be a system with basis $|0\rangle, |1\rangle, \ldots, |m - 1\rangle$. 
Measurement

The Born Rule

- Let $A$ be a system with basis $|0\rangle, |1\rangle, \ldots, |m-1\rangle$.
- Let $\psi = \sum_{i=0}^{m-1} a_i |i\rangle$ be a state of $A$. 
The Born Rule

- Let $A$ be a system with basis $|0\rangle, |1\rangle, \ldots, |m-1\rangle$.
- Let $\psi = \sum_{i=0}^{m-1} a_i |i\rangle$ be a state of $A$.
- We measure $A$ at this state.
The Born Rule

- Let $A$ be a system with basis $|0\rangle, |1\rangle, \ldots, |m - 1\rangle$.
- Let $\psi = \sum_{i=0}^{m-1} a_i |i\rangle$ be a state of $A$.
- We measure $A$ at this state.
- The output we get is one of the classical states $|0\rangle, |1\rangle, \ldots, |m - 1\rangle$. 

Public-key Cryptography: Theory and Practice
Abhijit Das
The Born Rule

- Let $A$ be a system with basis $|0\rangle, |1\rangle, \ldots, |m-1\rangle$.
- Let $\psi = \sum_{i=0}^{m-1} a_i |i\rangle$ be a state of $A$.
- We measure $A$ at this state.
- The output we get is one of the classical states $|0\rangle, |1\rangle, \ldots, |m-1\rangle$.
- The probability of observing $|i\rangle$ is $a_i^2$. 
Measurement

The Born Rule

- Let $A$ be a system with basis $|0\rangle, |1\rangle, \ldots, |m - 1\rangle$.
- Let $\psi = \sum_{i=0}^{m-1} a_i |i\rangle$ be a state of $A$.
- We measure $A$ at this state.
- The output we get is one of the classical states $|0\rangle, |1\rangle, \ldots, |m - 1\rangle$.
- The probability of observing $|i\rangle$ is $a_i^2$.
- If the outcome is $i$, the system collapses to the state $|i\rangle$. 
The Born Rule

- Let $A$ be a system with basis $|0\rangle, |1\rangle, \ldots, |m-1\rangle$.
- Let $\psi = \sum_{i=0}^{m-1} a_i |i\rangle$ be a state of $A$.
- We measure $A$ at this state.
- The output we get is one of the classical states $|0\rangle, |1\rangle, \ldots, |m-1\rangle$.
- The probability of observing $|i\rangle$ is $a_i^2$.
- If the outcome is $i$, the system collapses to the state $|i\rangle$.
- Measurement is, therefore, non-invertible.
Measurement

The Born Rule

- Let $A$ be a system with basis $|0\rangle, |1\rangle, \ldots, |m-1\rangle$.
- Let $\psi = \sum_{i=0}^{m-1} a_i |i\rangle$ be a state of $A$.
- We measure $A$ at this state.
- The output we get is one of the classical states $|0\rangle, |1\rangle, \ldots, |m-1\rangle$.
- The probability of observing $|i\rangle$ is $a_i^2$.
- If the outcome is $i$, the system collapses to the state $|i\rangle$.
- Measurement is, therefore, non-invertible.
- Measurement is often used to initialize a system.
Measurement

The Born Rule

- Let $A$ be a system with basis $|0\rangle, |1\rangle, \ldots, |m-1\rangle$.
- Let $\psi = \sum_{i=0}^{m-1} a_i|i\rangle$ be a state of $A$.
- We measure $A$ at this state.
- The output we get is one of the classical states $|0\rangle, |1\rangle, \ldots, |m-1\rangle$.
- The probability of observing $|i\rangle$ is $a_i^2$.
- If the outcome is $i$, the system collapses to the state $|i\rangle$.
- Measurement is, therefore, non-invertible.
- Measurement is often used to initialize a system.
- So sad! You cannot see Schrödinger’s cat in the state $\frac{1}{\sqrt{2}} (|\text{Alive}\rangle + |\text{Dead}\rangle)$.
The Generalized Born Rule
The Generalized Born Rule

Let $R$ be an $(m + n)$-bit quantum register in the state

$$|\psi\rangle_{m+n} = \sum_{i,j} a_{i,j} |i,j\rangle_{m+n} \text{ with } \sum_{i,j} |a_{i,j}|^2 = 1.$$
The Generalized Born Rule

- Let $R$ be an $(m + n)$-bit quantum register in the state
  $$|\psi\rangle_{m+n} = \sum_{i,j} a_{i,j} |i,j\rangle_{m+n} \text{ with } \sum_{i,j} |a_{i,j}|^2 = 1.$$ 

- We measure the left $m$ bits of $R$. 
The Generalized Born Rule

Let $R$ be an $(m + n)$-bit quantum register in the state

$$|\psi\rangle_{m+n} = \sum_{i,j} a_{i,j} |i,j\rangle_{m+n}$$

with $\sum_{i,j} |a_{i,j}|^2 = 1$.

We measure the left $m$ bits of $R$.

The outcome is an integer $i \in \{0, 1, 2, \ldots, 2^m - 1\}$ with probability

$$p_i = \sum_{j=0}^{2^n-1} |a_{i,j}|^2.$$
The Generalized Born Rule

- Let $R$ be an $(m + n)$-bit quantum register in the state
  \[ |\psi\rangle_{m+n} = \sum_{i,j} a_{i,j} |i,j\rangle_{m+n} \text{ with } \sum_{i,j} |a_{i,j}|^2 = 1. \]
- We measure the left $m$ bits of $R$.
- The outcome is an integer $i \in \{0, 1, 2, \ldots, 2^m - 1\}$ with probability
  \[ p_i = \sum_{j=0}^{2^n-1} |a_{i,j}|^2. \]
- $R$ collapses to the state $|i\rangle \left(\frac{1}{\sqrt{p_i}} \sum_j a_{i,j} |j\rangle_n\right)$. 
The Generalized Born Rule

Let $R$ be an $(m + n)$-bit quantum register in the state

$$|\psi\rangle_{m+n} = \sum_{i,j} a_{i,j} |i,j\rangle_{m+n}$$

with $\sum_{i,j} |a_{i,j}|^2 = 1$.

- We measure the left $m$ bits of $R$.
- The outcome is an integer $i \in \{0, 1, 2, \ldots, 2^m - 1\}$ with probability $p_i = \sum_{j=0}^{2^n-1} |a_{i,j}|^2$.

- $R$ collapses to the state $|i\rangle \left( \frac{1}{\sqrt{p_i}} \sum_j a_{i,j} |j\rangle_n \right)$.

- If we now measure the right $n$ bits, we get an integer $j \in \{0, 1, 2, \ldots, 2^n - 1\}$ with probability $|a_{i,j}|^2 / p_i$. 
The Generalized Born Rule

Let $R$ be an $(m + n)$-bit quantum register in the state

$$|\psi\rangle_{m+n} = \sum_{i,j} a_{i,j} |i,j\rangle_{m+n}$$

with $\sum_{i,j} |a_{i,j}|^2 = 1$.

We measure the left $m$ bits of $R$.

The outcome is an integer $i \in \{0, 1, 2, \ldots , 2^m - 1\}$ with

$$p_i = \sum_{j=0}^{2^n-1} |a_{i,j}|^2.$$

$R$ collapses to the state $|i\rangle \left( \frac{1}{\sqrt{p_i}} \sum_j a_{i,j} |j\rangle_n \right)$.

If we now measure the right $n$ bits, we get an integer $j \in \{0, 1, 2, \ldots , 2^n - 1\}$ with probability $|a_{i,j}|^2 / p_i$.

Probability of measuring $|i\rangle_m |j\rangle_n$ is $p_i |a_{i,j}|^2 / p_i = |a_{i,j}|^2$. 

Public-key Cryptography: Theory and Practice
Abhijit Das
A Computational Framework
A Computational Framework

- The input is an $m$-bit value $x$. 
A Computational Framework

- The input is an $m$-bit value $x$.
- We want to compute an $n$-bit value $f(x)$. 
The input is an $m$-bit value $x$.

We want to compute an $n$-bit value $f(x)$.

Even if $m = n$, the function $f$ need not be invertible.
A Computational Framework

- The input is an $m$-bit value $x$.
- We want to compute an $n$-bit value $f(x)$.
- Even if $m = n$, the function $f$ need not be invertible.
- Use an $(m + n)$-bit quantum register $R$. 
The input is an \( m \)-bit value \( x \).

We want to compute an \( n \)-bit value \( f(x) \).

Even if \( m = n \), the function \( f \) need not be invertible.

Use an \((m + n)\)-bit quantum register \( R \).

Initialize \( R \) to \( |x\rangle_m |0\rangle_n \).
The input is an \( m \)-bit value \( x \).

We want to compute an \( n \)-bit value \( f(x) \).

Even if \( m = n \), the function \( f \) need not be invertible.

Use an \(( m + n )\)-bit quantum register \( R \).

Initialize \( R \) to \(| x \rangle_m |0 \rangle_n \).

Apply the transformation \( U_f | x \rangle_m |y \rangle_n = | x \rangle_m |f(x) \oplus y \rangle_n \) on \( R \).
A Computational Framework

- The input is an $m$-bit value $x$.
- We want to compute an $n$-bit value $f(x)$.
- Even if $m = n$, the function $f$ need not be invertible.

- Use an $(m + n)$-bit quantum register $R$.
- Initialize $R$ to $|x\rangle_m|0\rangle_n$.
- Apply the transformation $U_f|x\rangle_m|y\rangle_n = |x\rangle_m|f(x) \oplus y\rangle_n$ on $R$.
- For $y = 0$, the output is $|x\rangle_m|f(x)\rangle_n$. 
The input is an $m$-bit value $x$.

We want to compute an $n$-bit value $f(x)$.

Even if $m = n$, the function $f$ need not be invertible.

Use an $(m + n)$-bit quantum register $R$.

Initialize $R$ to $|x\rangle_m|0\rangle_n$.

Apply the transformation $U_f|x\rangle_m|y\rangle_n = |x\rangle_m|f(x) \oplus y\rangle_n$ on $R$.

For $y = 0$, the output is $|x\rangle_m|f(x)\rangle_n$.

$U_f$ is a unitary transformation.
A Computational Framework

- The input is an \( m \)-bit value \( x \).
- We want to compute an \( n \)-bit value \( f(x) \).
- Even if \( m = n \), the function \( f \) need not be invertible.

- Use an \((m+n)\)-bit quantum register \( R \).
- Initialize \( R \) to \(|x\rangle_m |0\rangle_n \).
- Apply the transformation \( U_f |x\rangle_m |y\rangle_n = |x\rangle_m |f(x) \oplus y\rangle_n \) on \( R \).
- For \( y = 0 \), the output is \(|x\rangle_m |f(x)\rangle_n \).
- \( U_f \) is a unitary transformation.
- \( U_f^{-1} = U_f \).
The Deutsch Algorithm

\( f : \{0, 1\} \rightarrow \{0, 1\} \) is a function provided as a black box. We want to check whether \( f \) is a constant function (\( f(0) = f(1) \)).
The Deutsch Algorithm

\[ f : \{0, 1\} \rightarrow \{0, 1\} \] is a function provided as a black box. We want to check whether \( f \) is a constant function \((f(0) = f(1))\).

- Classical computation needs two invocations of the black box.
The Deutsch Algorithm

\[ f : \{0, 1\} \rightarrow \{0, 1\} \] is a function provided as a black box. We want to check whether \( f \) is a constant function (\( f(0) = f(1) \)).

- Classical computation needs two invocations of the black box.
- Quantum computation can achieve the same with one invocation only.
The Deutsch Algorithm

\[ f : \{0, 1\} \rightarrow \{0, 1\} \] is a function provided as a black box. We want to check whether \( f \) is a constant function (\( f(0) = f(1) \)).

- Classical computation needs two invocations of the black box.
- Quantum computation can achieve the same with one invocation only.
- Use a 2-bit register \( R \) (\( m = n = 1 \)).
The Deutsch Algorithm

\[ f : \{0, 1\} \rightarrow \{0, 1\} \text{ is a function provided as a black box.} \]
We want to check whether \( f \) is a constant function \((f(0) = f(1))\).

- Classical computation needs two invocations of the black box.
- Quantum computation can achieve the same with one invocation only.

- Use a 2-bit register \( R \ (m = n = 1) \).
- Use the unitary transform \( D_f |x\rangle |y\rangle = |x\rangle |f(x) \oplus y\rangle \).
The Deutsch Algorithm

\( f : \{0, 1\} \rightarrow \{0, 1\} \) is a function provided as a black box. We want to check whether \( f \) is a constant function \((f(0) = f(1))\).

- Classical computation needs two invocations of the black box.
- Quantum computation can achieve the same with one invocation only.

- Use a 2-bit register \( R \) \((m = n = 1)\).
- Use the unitary transform \( D_f |x\rangle|y\rangle = |x\rangle|f(x) \oplus y\rangle \).

- Initialize \( R \) to the state \( \frac{1}{\sqrt{2}} (|0\rangle - |1\rangle) \left( \frac{1}{\sqrt{2}} |0\rangle - \frac{1}{\sqrt{2}} |1\rangle \right) = \frac{1}{2} \left( |0\rangle|0\rangle - |0\rangle|1\rangle - |1\rangle|0\rangle + |1\rangle|1\rangle \right) \).
The Deutsch Algorithm (contd)
Applying $D_f$ on $R$ changes its state to

$$
\begin{cases} 
\frac{1}{2} (|0\rangle - |1\rangle) \left( |f(0)\rangle - |\bar{f}(0)\rangle \right) & \text{if } f(0) = f(1), \\
\frac{1}{2} (|0\rangle + |1\rangle) \left( |f(0)\rangle - |\bar{f}(0)\rangle \right) & \text{if } f(0) \neq f(1).
\end{cases}
$$
Applying $D_f$ on $R$ changes its state to

$$\begin{cases} 
\frac{1}{2} (|0\rangle - |1\rangle) (|f(0)\rangle - |\bar{f}(0)\rangle) & \text{if } f(0) = f(1), \\
\frac{1}{2} (|0\rangle + |1\rangle) (|f(0)\rangle - |\bar{f}(0)\rangle) & \text{if } f(0) \neq f(1).
\end{cases}$$

Apply the Hadamard transform on the left bit to change $R$ to the state

$$\begin{cases} 
|1\rangle \frac{1}{\sqrt{2}} (|f(0)\rangle - |\bar{f}(0)\rangle) & \text{if } f(0) = f(1), \\
|0\rangle \frac{1}{\sqrt{2}} (|f(0)\rangle - |\bar{f}(0)\rangle) & \text{if } f(0) \neq f(1).
\end{cases}$$
The Deutsch Algorithm (contd)

- Applying $D_f$ on $R$ changes its state to

$$\begin{cases} 
\frac{1}{2} (|0\rangle - |1\rangle) (|f(0)\rangle - |\overline{f}(0)\rangle) & \text{if } f(0) = f(1), \\
\frac{1}{2} (|0\rangle + |1\rangle) (|f(0)\rangle - |\overline{f}(0)\rangle) & \text{if } f(0) \neq f(1).
\end{cases}$$

- Apply the Hadamard transform on the left bit to change $R$ to the state

$$\begin{cases} 
|1\rangle \frac{1}{\sqrt{2}} (|f(0)\rangle - |\overline{f}(0)\rangle) & \text{if } f(0) = f(1), \\
|0\rangle \frac{1}{\sqrt{2}} (|f(0)\rangle - |\overline{f}(0)\rangle) & \text{if } f(0) \neq f(1).
\end{cases}$$

- Measure the left bit.
Applying $D_f$ on $R$ changes its state to

\[
\begin{cases}
    \frac{1}{2} (|0\rangle - |1\rangle) (|f(0)\rangle - |\bar{f}(0)\rangle) & \text{if } f(0) = f(1), \\
    \frac{1}{2} (|0\rangle + |1\rangle) (|f(0)\rangle - |\bar{f}(0)\rangle) & \text{if } f(0) \neq f(1).
\end{cases}
\]

Apply the Hadamard transform on the left bit to change $R$ to the state

\[
\begin{cases}
    |1\rangle \frac{1}{\sqrt{2}} (|f(0)\rangle - |\bar{f}(0)\rangle) & \text{if } f(0) = f(1), \\
    |0\rangle \frac{1}{\sqrt{2}} (|f(0)\rangle - |\bar{f}(0)\rangle) & \text{if } f(0) \neq f(1).
\end{cases}
\]

Measure the left bit.

The outcome is 1 or 0 according as whether $f$ is constant or not.
Quantum Key Exchange

The BB84 Protocol (Charles H. Bennett and Gilles Brassard, 1984)
Quantum Key Exchange

The BB84 Protocol (Charles H. Bennett and Gilles Brassard, 1984)

- Alice and Bob want to agree upon a secret key over an insecure channel.
Quantum Key Exchange

The BB84 Protocol (Charles H. Bennett and Gilles Brassard, 1984)

Alice and Bob want to agree upon a secret key over an insecure channel.

Alice sends a qubit to Bob
Quantum Key Exchange

The BB84 Protocol (Charles H. Bennett and Gilles Brassard, 1984)

- Alice and Bob want to agree upon a secret key over an insecure channel.

Alice sends a qubit to Bob
- Alice generates a random classical bit $i$. 
Quantum Key Exchange

The **BB84 Protocol** (Charles H. Bennett and Gilles Brassard, 1984)
- Alice and Bob want to agree upon a secret key over an insecure channel.

**Alice sends a qubit to Bob**
- Alice generates a random classical bit $i$.
- Alice makes a random decision $x$. 
Quantum Key Exchange

The BB84 Protocol (Charles H. Bennett and Gilles Brassard, 1984)

- Alice and Bob want to agree upon a secret key over an insecure channel.

Alice sends a qubit to Bob

- Alice generates a random classical bit $i$.
- Alice makes a random decision $x$.
- If $x = 0$, Alice sends the qubit $|i\rangle$ itself to Bob.
Quantum Key Exchange

The BB84 Protocol (Charles H. Bennett and Gilles Brassard, 1984)

- Alice and Bob want to agree upon a secret key over an insecure channel.

Alice sends a qubit to Bob

- Alice generates a random classical bit \( i \).
- Alice makes a random decision \( x \).
- If \( x = 0 \), Alice sends the qubit \( |i\rangle \) itself to Bob.
- If \( x = 1 \), Alice uses the Hadamard transform and sends \( H|i\rangle \) (\( H|0\rangle = \frac{1}{\sqrt{2}}(|0\rangle + |1\rangle) \) or \( H|1\rangle = \frac{1}{\sqrt{2}}(|0\rangle - |1\rangle) \)) to Bob.
The BB84 Algorithm (contd)
The BB84 Algorithm (contd)

Bob processes Alice’s qubit
The BB84 Algorithm (contd)

Bob processes Alice’s qubit

Let $A$ be the qubit received by Bob from Alice.
Bob processes Alice’s qubit

- Let $A$ be the qubit received by Bob from Alice.
- Bob makes a random guess $y$ about Alice’s decision $x$. 
Bob processes Alice’s qubit
- Let $A$ be the qubit received by Bob from Alice.
- Bob makes a random guess $y$ about Alice’s decision $x$.
- If $y = 0$, Bob takes $B = A$. 
Bob processes Alice’s qubit

- Let $A$ be the qubit received by Bob from Alice.
- Bob makes a random guess $y$ about Alice’s decision $x$.
- If $y = 0$, Bob takes $B = A$.
- If $y = 1$, Bob applies the Hadamard transform to compute $B = HA$.  


The BB84 Algorithm (contd)

**Bob processes Alice’s qubit**

- Let $A$ be the qubit received by Bob from Alice.
- Bob makes a random guess $y$ about Alice’s decision $x$.
- If $y = 0$, Bob takes $B = A$.
- If $y = 1$, Bob applies the Hadamard transform to compute $B = HA$.
- Bob measures $B$ to obtain the classical bit $j$. 
Bob processes Alice’s qubit

- Let $A$ be the qubit received by Bob from Alice.
- Bob makes a random guess $y$ about Alice’s decision $x$.
- If $y = 0$, Bob takes $B = A$.
- If $y = 1$, Bob applies the Hadamard transform to compute $B = HA$.
- Bob measures $B$ to obtain the classical bit $j$.

Alice and Bob exchange their guesses
The BB84 Algorithm (contd)

Bob processes Alice’s qubit

- Let $A$ be the qubit received by Bob from Alice.
- Bob makes a random guess $y$ about Alice’s decision $x$.
- If $y = 0$, Bob takes $B = A$.
- If $y = 1$, Bob applies the Hadamard transform to compute $B = HA$.
- Bob measures $B$ to obtain the classical bit $j$.

Alice and Bob exchange their guesses

- Bob sends $y$ to Alice.
Bob processes Alice’s qubit

- Let $A$ be the qubit received by Bob from Alice.
- Bob makes a random guess $y$ about Alice’s decision $x$.
- If $y = 0$, Bob takes $B = A$.
- If $y = 1$, Bob applies the Hadamard transform to compute $B = HA$.
- Bob measures $B$ to obtain the classical bit $j$.

Alice and Bob exchange their guesses

- Bob sends $y$ to Alice.
- Alice sends $x$ to Bob.
The BB84 Algorithm (contd)

Bob processes Alice’s qubit
- Let \( A \) be the qubit received by Bob from Alice.
- Bob makes a random guess \( y \) about Alice’s decision \( x \).
- If \( y = 0 \), Bob takes \( B = A \).
- If \( y = 1 \), Bob applies the Hadamard transform to compute \( B = HA \).
- Bob measures \( B \) to obtain the classical bit \( j \).

Alice and Bob exchange their guesses
- Bob sends \( y \) to Alice.
- Alice sends \( x \) to Bob.
- If \( x = y \), Alice and Bob store the common bit \( i = j \).
The BB84 Algorithm: Correctness
The BB84 Algorithm: Correctness

If $x = y = 0$, then Alice sends $A = |i\rangle$ to Bob, and Bob measures $B = A = |i\rangle$ to obtain $j = i$. 
The BB84 Algorithm: Correctness

- If $x = y = 0$, then Alice sends $A = |i\rangle$ to Bob, and Bob measures $B = A = |i\rangle$ to obtain $j = i$.
- If $x = y = 1$, then Alice sends $A = H|i\rangle$ to Bob, and Bob computes $B = HA = H^2|i\rangle = |i\rangle$. Measurement gives $j = i$. 
The BB84 Algorithm: Correctness

- If \( x = y = 0 \), then Alice sends \( A = |i\rangle \) to Bob, and Bob measures \( B = A = |i\rangle \) to obtain \( j = i \).
- If \( x = y = 1 \), then Alice sends \( A = H|i\rangle \) to Bob, and Bob computes \( B = HA = H^2|i\rangle = |i\rangle \). Measurement gives \( j = i \).
- If \( x = 0 \) and \( y = 1 \) or if \( x = 1 \) and \( y = 0 \), then \( B = H|i\rangle \), so measurement reveals 0 or 1, each with probability 1/2.
The BB84 Algorithm: Correctness

- If $x = y = 0$, then Alice sends $A = |i\rangle$ to Bob, and Bob measures $B = A = |i\rangle$ to obtain $j = i$.
- If $x = y = 1$, then Alice sends $A = H|i\rangle$ to Bob, and Bob computes $B = HA = H^2|i\rangle = |i\rangle$. Measurement gives $j = i$.
- If $x = 0$ and $y = 1$ or if $x = 1$ and $y = 0$, then $B = H|i\rangle$, so measurement reveals 0 or 1, each with probability $1/2$.
- Now, $j$ gives no clue about $i$. 

Public-key Cryptography: Theory and Practice

Abhijit Das
The BB84 Algorithm: Correctness

- If $x = y = 0$, then Alice sends $A = |i\rangle$ to Bob, and Bob measures $B = A = |i\rangle$ to obtain $j = i$.
- If $x = y = 1$, then Alice sends $A = H|i\rangle$ to Bob, and Bob computes $B = HA = H^2|i\rangle = |i\rangle$. Measurement gives $j = i$.
- If $x = 0$ and $y = 1$ or if $x = 1$ and $y = 0$, then $B = H|i\rangle$, so measurement reveals 0 or 1, each with probability $1/2$.
- Now, $j$ gives no clue about $i$.
- Alice and Bob discard $i$ and $j$. 
The BB84 Algorithm: Correctness

- If $x = y = 0$, then Alice sends $A = |i\rangle$ to Bob, and Bob measures $B = A = |i\rangle$ to obtain $j = i$.
- If $x = y = 1$, then Alice sends $A = H|i\rangle$ to Bob, and Bob computes $B = HA = H^2|i\rangle = |i\rangle$. Measurement gives $j = i$.
- If $x = 0$ and $y = 1$ or if $x = 1$ and $y = 0$, then $B = H|i\rangle$, so measurement reveals 0 or 1, each with probability $1/2$.
- Now, $j$ gives no clue about $i$.
- Alice and Bob discard $i$ and $j$.
- About half of the time, Alice and Bob make the same independent guess $x = y$. 
The BB84 Algorithm: Correctness

- If $x = y = 0$, then Alice sends $A = |i\rangle$ to Bob, and Bob measures $B = A = |i\rangle$ to obtain $j = i$.
- If $x = y = 1$, then Alice sends $A = H|i\rangle$ to Bob, and Bob computes $B = HA = H^2|i\rangle = |i\rangle$. Measurement gives $j = i$.
- If $x = 0$ and $y = 1$ or if $x = 1$ and $y = 0$, then $B = H|i\rangle$, so measurement reveals 0 or 1, each with probability $1/2$.
- Now, $j$ gives no clue about $i$.
- Alice and Bob discard $i$ and $j$.

- About half of the time, Alice and Bob make the same independent guess $x = y$.
- In about $2^n$ iterations, a common $n$-bit key can be established.
### The BB84 Algorithm: Example

<table>
<thead>
<tr>
<th>Iteration</th>
<th>$i$</th>
<th>$x$</th>
<th>$A$</th>
<th>$y$</th>
<th>$B$</th>
<th>$j$</th>
<th>Common bit</th>
</tr>
</thead>
</table>

The BB84 Algorithm is a protocol for quantum key distribution. It allows two parties to establish a shared secret key over an insecure channel. The example table provides a simple demonstration of how the algorithm works, showing the iteration, exchanged bits, and the common bit obtained after reconciliation and privacy amplification.
The BB84 Algorithm: Example

<table>
<thead>
<tr>
<th>Iteration</th>
<th>$i$</th>
<th>$x$</th>
<th>$A$</th>
<th>$y$</th>
<th>$B$</th>
<th>$j$</th>
<th>Common bit</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>$\frac{1}{\sqrt{2}}(</td>
<td>0\rangle +</td>
<td>1\rangle)$</td>
<td>0</td>
<td>$\frac{1}{\sqrt{2}}(</td>
</tr>
</tbody>
</table>
### The BB84 Algorithm: Example

<table>
<thead>
<tr>
<th>Iteration</th>
<th>$i$</th>
<th>$x$</th>
<th>$A$</th>
<th>$y$</th>
<th>$B$</th>
<th>$j$</th>
<th>Common bit</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>$\frac{1}{\sqrt{2}}(</td>
<td>0\rangle +</td>
<td>1\rangle)$</td>
<td>0</td>
<td>$\frac{1}{\sqrt{2}}(</td>
</tr>
<tr>
<td>2</td>
<td>0</td>
<td>0</td>
<td>$</td>
<td>0\rangle$</td>
<td>1</td>
<td>$\frac{1}{\sqrt{2}}(</td>
<td>0\rangle +</td>
</tr>
</tbody>
</table>
### The BB84 Algorithm: Example

<table>
<thead>
<tr>
<th>Iteration</th>
<th>( i )</th>
<th>( x )</th>
<th>( A )</th>
<th>( y )</th>
<th>( B )</th>
<th>( j )</th>
<th>Common bit</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>( \frac{1}{\sqrt{2}}(</td>
<td>0\rangle +</td>
<td>1\rangle) )</td>
<td>0</td>
<td>( \frac{1}{\sqrt{2}}(</td>
</tr>
<tr>
<td>2</td>
<td>0</td>
<td>0</td>
<td>(</td>
<td>0\rangle )</td>
<td>1</td>
<td>( \frac{1}{\sqrt{2}}(</td>
<td>0\rangle +</td>
</tr>
<tr>
<td>3</td>
<td>0</td>
<td>1</td>
<td>( \frac{1}{\sqrt{2}}(</td>
<td>0\rangle +</td>
<td>1\rangle) )</td>
<td>1</td>
<td>(</td>
</tr>
</tbody>
</table>
### The BB84 Algorithm: Example

<table>
<thead>
<tr>
<th>Iteration</th>
<th>$i$</th>
<th>$x$</th>
<th>$A$</th>
<th>$y$</th>
<th>$B$</th>
<th>$j$</th>
<th>Common bit</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>$\frac{1}{\sqrt{2}}(</td>
<td>0\rangle +</td>
<td>1\rangle)$</td>
<td>0</td>
<td>$\frac{1}{\sqrt{2}}(</td>
</tr>
<tr>
<td>2</td>
<td>0</td>
<td>0</td>
<td>$</td>
<td>0\rangle$</td>
<td>1</td>
<td>$\frac{1}{\sqrt{2}}(</td>
<td>0\rangle +</td>
</tr>
<tr>
<td>3</td>
<td>0</td>
<td>1</td>
<td>$\frac{1}{\sqrt{2}}(</td>
<td>0\rangle +</td>
<td>1\rangle)$</td>
<td>1</td>
<td>$</td>
</tr>
<tr>
<td>4</td>
<td>1</td>
<td>1</td>
<td>$\frac{1}{\sqrt{2}}(</td>
<td>0\rangle -</td>
<td>1\rangle)$</td>
<td>0</td>
<td>$\frac{1}{\sqrt{2}}(</td>
</tr>
</tbody>
</table>
# The BB84 Algorithm: Example

<table>
<thead>
<tr>
<th>Iteration</th>
<th>$i$</th>
<th>$x$</th>
<th>$A$</th>
<th>$y$</th>
<th>$B$</th>
<th>$j$</th>
<th>Common bit</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>$\frac{1}{\sqrt{2}}(</td>
<td>0\rangle +</td>
<td>1\rangle)$</td>
<td>0</td>
<td>$\frac{1}{\sqrt{2}}(</td>
</tr>
<tr>
<td>2</td>
<td>0</td>
<td>0</td>
<td>$</td>
<td>0\rangle$</td>
<td>1</td>
<td>$\frac{1}{\sqrt{2}}(</td>
<td>0\rangle +</td>
</tr>
<tr>
<td>3</td>
<td>0</td>
<td>1</td>
<td>$\frac{1}{\sqrt{2}}(</td>
<td>0\rangle +</td>
<td>1\rangle)$</td>
<td>1</td>
<td>$</td>
</tr>
<tr>
<td>4</td>
<td>1</td>
<td>1</td>
<td>$\frac{1}{\sqrt{2}}(</td>
<td>0\rangle -</td>
<td>1\rangle)$</td>
<td>0</td>
<td>$\frac{1}{\sqrt{2}}(</td>
</tr>
<tr>
<td>5</td>
<td>0</td>
<td>0</td>
<td>$</td>
<td>0\rangle$</td>
<td>0</td>
<td>$</td>
<td>0\rangle$</td>
</tr>
</tbody>
</table>
The BB84 Algorithm: Example

<table>
<thead>
<tr>
<th>Iteration</th>
<th>$i$</th>
<th>$x$</th>
<th>$A$</th>
<th>$y$</th>
<th>$B$</th>
<th>$j$</th>
<th>Common bit</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>$\frac{1}{\sqrt{2}}(</td>
<td>0\rangle +</td>
<td>1\rangle)$</td>
<td>0</td>
<td>$\frac{1}{\sqrt{2}}(</td>
</tr>
<tr>
<td>2</td>
<td>0</td>
<td>0</td>
<td>$</td>
<td>0\rangle$</td>
<td>1</td>
<td>$\frac{1}{\sqrt{2}}(</td>
<td>0\rangle +</td>
</tr>
<tr>
<td>3</td>
<td>0</td>
<td>1</td>
<td>$\frac{1}{\sqrt{2}}(</td>
<td>0\rangle +</td>
<td>1\rangle)$</td>
<td>1</td>
<td>$</td>
</tr>
<tr>
<td>4</td>
<td>1</td>
<td>1</td>
<td>$\frac{1}{\sqrt{2}}(</td>
<td>0\rangle -</td>
<td>1\rangle)$</td>
<td>0</td>
<td>$\frac{1}{\sqrt{2}}(</td>
</tr>
<tr>
<td>5</td>
<td>0</td>
<td>0</td>
<td>$</td>
<td>0\rangle$</td>
<td>0</td>
<td>$</td>
<td>0\rangle$</td>
</tr>
<tr>
<td>6</td>
<td>1</td>
<td>1</td>
<td>$\frac{1}{\sqrt{2}}(</td>
<td>0\rangle -</td>
<td>1\rangle)$</td>
<td>1</td>
<td>$</td>
</tr>
</tbody>
</table>
## The BB84 Algorithm: Example

<table>
<thead>
<tr>
<th>Iteration</th>
<th>$i$</th>
<th>$x$</th>
<th>$A$</th>
<th>$y$</th>
<th>$B$</th>
<th>$j$</th>
<th>Common bit</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>$\frac{1}{\sqrt{2}}(</td>
<td>0\rangle +</td>
<td>1\rangle)$</td>
<td>0</td>
<td>$\frac{1}{\sqrt{2}}(</td>
</tr>
<tr>
<td>2</td>
<td>0</td>
<td>0</td>
<td>$</td>
<td>0\rangle$</td>
<td>1</td>
<td>$\frac{1}{\sqrt{2}}(</td>
<td>0\rangle +</td>
</tr>
<tr>
<td>3</td>
<td>0</td>
<td>1</td>
<td>$\frac{1}{\sqrt{2}}(</td>
<td>0\rangle +</td>
<td>1\rangle)$</td>
<td>1</td>
<td>$</td>
</tr>
<tr>
<td>4</td>
<td>1</td>
<td>1</td>
<td>$\frac{1}{\sqrt{2}}(</td>
<td>0\rangle -</td>
<td>1\rangle)$</td>
<td>0</td>
<td>$\frac{1}{\sqrt{2}}(</td>
</tr>
<tr>
<td>5</td>
<td>0</td>
<td>0</td>
<td>$</td>
<td>0\rangle$</td>
<td>0</td>
<td>$</td>
<td>0\rangle$</td>
</tr>
<tr>
<td>6</td>
<td>1</td>
<td>1</td>
<td>$\frac{1}{\sqrt{2}}(</td>
<td>0\rangle -</td>
<td>1\rangle)$</td>
<td>1</td>
<td>$</td>
</tr>
<tr>
<td>7</td>
<td>1</td>
<td>1</td>
<td>$\frac{1}{\sqrt{2}}(</td>
<td>0\rangle -</td>
<td>1\rangle)$</td>
<td>0</td>
<td>$\frac{1}{\sqrt{2}}(</td>
</tr>
</tbody>
</table>
### The BB84 Algorithm: Example

<table>
<thead>
<tr>
<th>Iteration</th>
<th>$i$</th>
<th>$x$</th>
<th>$A$</th>
<th>$y$</th>
<th>$B$</th>
<th>$j$</th>
<th>Common bit</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>(\frac{1}{\sqrt{2}}(</td>
<td>0\rangle +</td>
<td>1\rangle))</td>
<td>0</td>
<td>(\frac{1}{\sqrt{2}}(</td>
</tr>
<tr>
<td>2</td>
<td>0</td>
<td>0</td>
<td>(</td>
<td>0\rangle)</td>
<td>1</td>
<td>(\frac{1}{\sqrt{2}}(</td>
<td>0\rangle +</td>
</tr>
<tr>
<td>3</td>
<td>0</td>
<td>1</td>
<td>(\frac{1}{\sqrt{2}}(</td>
<td>0\rangle +</td>
<td>1\rangle))</td>
<td>1</td>
<td>(</td>
</tr>
<tr>
<td>4</td>
<td>1</td>
<td>1</td>
<td>(\frac{1}{\sqrt{2}}(</td>
<td>0\rangle -</td>
<td>1\rangle))</td>
<td>0</td>
<td>(\frac{1}{\sqrt{2}}(</td>
</tr>
<tr>
<td>5</td>
<td>0</td>
<td>0</td>
<td>(</td>
<td>0\rangle)</td>
<td>0</td>
<td>(</td>
<td>0\rangle)</td>
</tr>
<tr>
<td>6</td>
<td>1</td>
<td>1</td>
<td>(\frac{1}{\sqrt{2}}(</td>
<td>0\rangle -</td>
<td>1\rangle))</td>
<td>1</td>
<td>(</td>
</tr>
<tr>
<td>7</td>
<td>1</td>
<td>1</td>
<td>(\frac{1}{\sqrt{2}}(</td>
<td>0\rangle -</td>
<td>1\rangle))</td>
<td>0</td>
<td>(\frac{1}{\sqrt{2}}(</td>
</tr>
<tr>
<td>8</td>
<td>0</td>
<td>0</td>
<td>(</td>
<td>0\rangle)</td>
<td>0</td>
<td>(</td>
<td>0\rangle)</td>
</tr>
</tbody>
</table>
## The BB84 Algorithm: Example

<table>
<thead>
<tr>
<th>Iteration</th>
<th>$i$</th>
<th>$x$</th>
<th>$A$</th>
<th>$y$</th>
<th>$B$</th>
<th>$j$</th>
<th>Common bit</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>$\frac{1}{\sqrt{2}}(</td>
<td>0\rangle +</td>
<td>1\rangle)$</td>
<td>0</td>
<td>$\frac{1}{\sqrt{2}}(</td>
</tr>
<tr>
<td>2</td>
<td>0</td>
<td>0</td>
<td>$</td>
<td>0\rangle$</td>
<td>1</td>
<td>$\frac{1}{\sqrt{2}}(</td>
<td>0\rangle +</td>
</tr>
<tr>
<td>3</td>
<td>0</td>
<td>1</td>
<td>$\frac{1}{\sqrt{2}}(</td>
<td>0\rangle +</td>
<td>1\rangle)$</td>
<td>1</td>
<td>$</td>
</tr>
<tr>
<td>4</td>
<td>1</td>
<td>1</td>
<td>$\frac{1}{\sqrt{2}}(</td>
<td>0\rangle -</td>
<td>1\rangle)$</td>
<td>0</td>
<td>$\frac{1}{\sqrt{2}}(</td>
</tr>
<tr>
<td>5</td>
<td>0</td>
<td>0</td>
<td>$</td>
<td>0\rangle$</td>
<td>0</td>
<td>$</td>
<td>0\rangle$</td>
</tr>
<tr>
<td>6</td>
<td>1</td>
<td>1</td>
<td>$\frac{1}{\sqrt{2}}(</td>
<td>0\rangle -</td>
<td>1\rangle)$</td>
<td>1</td>
<td>$</td>
</tr>
<tr>
<td>7</td>
<td>1</td>
<td>1</td>
<td>$\frac{1}{\sqrt{2}}(</td>
<td>0\rangle -</td>
<td>1\rangle)$</td>
<td>0</td>
<td>$\frac{1}{\sqrt{2}}(</td>
</tr>
<tr>
<td>8</td>
<td>0</td>
<td>0</td>
<td>$</td>
<td>0\rangle$</td>
<td>0</td>
<td>$</td>
<td>0\rangle$</td>
</tr>
<tr>
<td>9</td>
<td>1</td>
<td>0</td>
<td>$</td>
<td>1\rangle$</td>
<td>1</td>
<td>$\frac{1}{\sqrt{2}}(</td>
<td>0\rangle -</td>
</tr>
</tbody>
</table>
## The BB84 Algorithm: Example

<table>
<thead>
<tr>
<th>Iteration</th>
<th>$i$</th>
<th>$x$</th>
<th>$A$</th>
<th>$y$</th>
<th>$B$</th>
<th>$j$</th>
<th>Common bit</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>$\frac{1}{\sqrt{2}}(</td>
<td>0\rangle +</td>
<td>1\rangle)$</td>
<td>0</td>
<td>$\frac{1}{\sqrt{2}}(</td>
</tr>
<tr>
<td>2</td>
<td>0</td>
<td>0</td>
<td>$</td>
<td>0\rangle$</td>
<td>1</td>
<td>$\frac{1}{\sqrt{2}}(</td>
<td>0\rangle +</td>
</tr>
<tr>
<td>3</td>
<td>0</td>
<td>1</td>
<td>$\frac{1}{\sqrt{2}}(</td>
<td>0\rangle +</td>
<td>1\rangle)$</td>
<td>1</td>
<td>$</td>
</tr>
<tr>
<td>4</td>
<td>1</td>
<td>1</td>
<td>$\frac{1}{\sqrt{2}}(</td>
<td>0\rangle -</td>
<td>1\rangle)$</td>
<td>0</td>
<td>$\frac{1}{\sqrt{2}}(</td>
</tr>
<tr>
<td>5</td>
<td>0</td>
<td>0</td>
<td>$</td>
<td>0\rangle$</td>
<td>0</td>
<td>$</td>
<td>0\rangle$</td>
</tr>
<tr>
<td>6</td>
<td>1</td>
<td>1</td>
<td>$\frac{1}{\sqrt{2}}(</td>
<td>0\rangle -</td>
<td>1\rangle)$</td>
<td>1</td>
<td>$</td>
</tr>
<tr>
<td>7</td>
<td>1</td>
<td>1</td>
<td>$\frac{1}{\sqrt{2}}(</td>
<td>0\rangle -</td>
<td>1\rangle)$</td>
<td>0</td>
<td>$\frac{1}{\sqrt{2}}(</td>
</tr>
<tr>
<td>8</td>
<td>0</td>
<td>0</td>
<td>$</td>
<td>0\rangle$</td>
<td>0</td>
<td>$</td>
<td>0\rangle$</td>
</tr>
<tr>
<td>9</td>
<td>1</td>
<td>0</td>
<td>$</td>
<td>1\rangle$</td>
<td>1</td>
<td>$\frac{1}{\sqrt{2}}(</td>
<td>0\rangle -</td>
</tr>
<tr>
<td>10</td>
<td>1</td>
<td>0</td>
<td>$</td>
<td>1\rangle$</td>
<td>0</td>
<td>$</td>
<td>1\rangle$</td>
</tr>
</tbody>
</table>
### The BB84 Algorithm: Example

<table>
<thead>
<tr>
<th>Iteration</th>
<th>$i$</th>
<th>$x$</th>
<th>$A$</th>
<th>$y$</th>
<th>$B$</th>
<th>$j$</th>
<th>Common bit</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>$\frac{1}{\sqrt{2}}(</td>
<td>0\rangle +</td>
<td>1\rangle)$</td>
<td>0</td>
<td>$\frac{1}{\sqrt{2}}(</td>
</tr>
<tr>
<td>2</td>
<td>0</td>
<td>0</td>
<td>$</td>
<td>0\rangle$</td>
<td>1</td>
<td>$\frac{1}{\sqrt{2}}(</td>
<td>0\rangle +</td>
</tr>
<tr>
<td>3</td>
<td>0</td>
<td>1</td>
<td>$\frac{1}{\sqrt{2}}(</td>
<td>0\rangle +</td>
<td>1\rangle)$</td>
<td>1</td>
<td>$</td>
</tr>
<tr>
<td>4</td>
<td>1</td>
<td>1</td>
<td>$\frac{1}{\sqrt{2}}(</td>
<td>0\rangle -</td>
<td>1\rangle)$</td>
<td>0</td>
<td>$\frac{1}{\sqrt{2}}(</td>
</tr>
<tr>
<td>5</td>
<td>0</td>
<td>0</td>
<td>$</td>
<td>0\rangle$</td>
<td>0</td>
<td>$</td>
<td>0\rangle$</td>
</tr>
<tr>
<td>6</td>
<td>1</td>
<td>1</td>
<td>$\frac{1}{\sqrt{2}}(</td>
<td>0\rangle -</td>
<td>1\rangle)$</td>
<td>1</td>
<td>$</td>
</tr>
<tr>
<td>7</td>
<td>1</td>
<td>1</td>
<td>$\frac{1}{\sqrt{2}}(</td>
<td>0\rangle -</td>
<td>1\rangle)$</td>
<td>0</td>
<td>$\frac{1}{\sqrt{2}}(</td>
</tr>
<tr>
<td>8</td>
<td>0</td>
<td>0</td>
<td>$</td>
<td>0\rangle$</td>
<td>0</td>
<td>$</td>
<td>0\rangle$</td>
</tr>
<tr>
<td>9</td>
<td>1</td>
<td>0</td>
<td>$</td>
<td>1\rangle$</td>
<td>1</td>
<td>$\frac{1}{\sqrt{2}}(</td>
<td>0\rangle -</td>
</tr>
<tr>
<td>10</td>
<td>1</td>
<td>0</td>
<td>$</td>
<td>1\rangle$</td>
<td>0</td>
<td>$</td>
<td>1\rangle$</td>
</tr>
<tr>
<td>11</td>
<td>0</td>
<td>0</td>
<td>$</td>
<td>0\rangle$</td>
<td>1</td>
<td>$\frac{1}{\sqrt{2}}(</td>
<td>0\rangle +</td>
</tr>
</tbody>
</table>
### The BB84 Algorithm: Example

<table>
<thead>
<tr>
<th>Iteration</th>
<th>$i$</th>
<th>$x$</th>
<th>$A$</th>
<th>$y$</th>
<th>$B$</th>
<th>$j$</th>
<th>Common bit</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>$\frac{1}{\sqrt{2}}(</td>
<td>0\rangle +</td>
<td>1\rangle)$</td>
<td>0</td>
<td>$\frac{1}{\sqrt{2}}(</td>
</tr>
<tr>
<td>2</td>
<td>0</td>
<td>0</td>
<td>$</td>
<td>0\rangle$</td>
<td>1</td>
<td>$\frac{1}{\sqrt{2}}(</td>
<td>0\rangle +</td>
</tr>
<tr>
<td>3</td>
<td>0</td>
<td>1</td>
<td>$\frac{1}{\sqrt{2}}(</td>
<td>0\rangle +</td>
<td>1\rangle)$</td>
<td>1</td>
<td>$</td>
</tr>
<tr>
<td>4</td>
<td>1</td>
<td>1</td>
<td>$\frac{1}{\sqrt{2}}(</td>
<td>0\rangle -</td>
<td>1\rangle)$</td>
<td>0</td>
<td>$\frac{1}{\sqrt{2}}(</td>
</tr>
<tr>
<td>5</td>
<td>0</td>
<td>0</td>
<td>$</td>
<td>0\rangle$</td>
<td>0</td>
<td>$</td>
<td>0\rangle$</td>
</tr>
<tr>
<td>6</td>
<td>1</td>
<td>1</td>
<td>$\frac{1}{\sqrt{2}}(</td>
<td>0\rangle -</td>
<td>1\rangle)$</td>
<td>1</td>
<td>$</td>
</tr>
<tr>
<td>7</td>
<td>1</td>
<td>1</td>
<td>$\frac{1}{\sqrt{2}}(</td>
<td>0\rangle -</td>
<td>1\rangle)$</td>
<td>0</td>
<td>$\frac{1}{\sqrt{2}}(</td>
</tr>
<tr>
<td>8</td>
<td>0</td>
<td>0</td>
<td>$</td>
<td>0\rangle$</td>
<td>0</td>
<td>$</td>
<td>0\rangle$</td>
</tr>
<tr>
<td>9</td>
<td>1</td>
<td>0</td>
<td>$</td>
<td>1\rangle$</td>
<td>1</td>
<td>$\frac{1}{\sqrt{2}}(</td>
<td>0\rangle -</td>
</tr>
<tr>
<td>10</td>
<td>1</td>
<td>0</td>
<td>$</td>
<td>1\rangle$</td>
<td>0</td>
<td>$</td>
<td>1\rangle$</td>
</tr>
<tr>
<td>11</td>
<td>0</td>
<td>0</td>
<td>$</td>
<td>0\rangle$</td>
<td>1</td>
<td>$\frac{1}{\sqrt{2}}(</td>
<td>0\rangle +</td>
</tr>
<tr>
<td>12</td>
<td>0</td>
<td>0</td>
<td>$</td>
<td>0\rangle$</td>
<td>1</td>
<td>$\frac{1}{\sqrt{2}}(</td>
<td>0\rangle +</td>
</tr>
</tbody>
</table>
## The BB84 Algorithm: Example

<table>
<thead>
<tr>
<th>Iteration</th>
<th>$i$</th>
<th>$x$</th>
<th>$A$</th>
<th>$y$</th>
<th>$B$</th>
<th>$j$</th>
<th>Common bit</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>$\frac{1}{\sqrt{2}}(</td>
<td>0\rangle +</td>
<td>1\rangle)$</td>
<td>0</td>
<td>$\frac{1}{\sqrt{2}}(</td>
</tr>
<tr>
<td>2</td>
<td>0</td>
<td>0</td>
<td>$</td>
<td>0\rangle$</td>
<td>1</td>
<td>$\frac{1}{\sqrt{2}}(</td>
<td>0\rangle +</td>
</tr>
<tr>
<td>3</td>
<td>0</td>
<td>1</td>
<td>$\frac{1}{\sqrt{2}}(</td>
<td>0\rangle +</td>
<td>1\rangle)$</td>
<td>1</td>
<td>$</td>
</tr>
<tr>
<td>4</td>
<td>1</td>
<td>1</td>
<td>$\frac{1}{\sqrt{2}}(</td>
<td>0\rangle -</td>
<td>1\rangle)$</td>
<td>0</td>
<td>$\frac{1}{\sqrt{2}}(</td>
</tr>
<tr>
<td>5</td>
<td>0</td>
<td>0</td>
<td>$</td>
<td>0\rangle$</td>
<td>0</td>
<td>$</td>
<td>0\rangle$</td>
</tr>
<tr>
<td>6</td>
<td>1</td>
<td>1</td>
<td>$\frac{1}{\sqrt{2}}(</td>
<td>0\rangle -</td>
<td>1\rangle)$</td>
<td>1</td>
<td>$</td>
</tr>
<tr>
<td>7</td>
<td>1</td>
<td>1</td>
<td>$\frac{1}{\sqrt{2}}(</td>
<td>0\rangle -</td>
<td>1\rangle)$</td>
<td>0</td>
<td>$\frac{1}{\sqrt{2}}(</td>
</tr>
<tr>
<td>8</td>
<td>0</td>
<td>0</td>
<td>$</td>
<td>0\rangle$</td>
<td>0</td>
<td>$</td>
<td>0\rangle$</td>
</tr>
<tr>
<td>9</td>
<td>1</td>
<td>0</td>
<td>$</td>
<td>1\rangle$</td>
<td>1</td>
<td>$\frac{1}{\sqrt{2}}(</td>
<td>0\rangle -</td>
</tr>
<tr>
<td>10</td>
<td>1</td>
<td>0</td>
<td>$</td>
<td>1\rangle$</td>
<td>0</td>
<td>$</td>
<td>1\rangle$</td>
</tr>
<tr>
<td>11</td>
<td>0</td>
<td>0</td>
<td>$</td>
<td>0\rangle$</td>
<td>1</td>
<td>$\frac{1}{\sqrt{2}}(</td>
<td>0\rangle +</td>
</tr>
<tr>
<td>12</td>
<td>0</td>
<td>0</td>
<td>$</td>
<td>0\rangle$</td>
<td>1</td>
<td>$\frac{1}{\sqrt{2}}(</td>
<td>0\rangle +</td>
</tr>
<tr>
<td>13</td>
<td>1</td>
<td>1</td>
<td>$\frac{1}{\sqrt{2}}(</td>
<td>0\rangle -</td>
<td>1\rangle)$</td>
<td>1</td>
<td>$</td>
</tr>
</tbody>
</table>
The BB84 Algorithm: Passive Eavesdropping
The BB84 Algorithm: Passive Eavesdropping

- Carol intercepts A.
The BB84 Algorithm: Passive Eavesdropping

- Carol intercepts $A$.
- Carol makes a guess $z$ about $x$. 
The BB84 Algorithm: Passive Eavesdropping

- Carol intercepts $A$.
- Carol makes a guess $z$ about $x$.
- If $z = 0$, Carol takes $C = A$, else Carol takes $C = HA$. 
The BB84 Algorithm: Passive Eavesdropping

- Carol intercepts $A$.
- Carol makes a guess $z$ about $x$.
- If $z = 0$, Carol takes $C = A$, else Carol takes $C = HA$.
- Carol measures $C$ to get the classical bit $k$. 
The BB84 Algorithm: Passive Eavesdropping

- Carol intercepts $A$.
- Carol makes a guess $z$ about $x$.
- If $z = 0$, Carol takes $C = A$, else Carol takes $C = HA$.
- Carol measures $C$ to get the classical bit $k$.
- Carol sends the measured qubit $D$ to Bob.
The BB84 Algorithm: Passive Eavesdropping

- Carol intercepts $A$.
- Carol makes a guess $z$ about $x$.
- If $z = 0$, Carol takes $C = A$, else Carol takes $C = HA$.
- Carol measures $C$ to get the classical bit $k$.
- Carol sends the measured qubit $D$ to Bob.
- Bob processes $D$ as if he has received $A$ from Alice.
The BB84 Algorithm: Passive Eavesdropping

- Carol intercepts $A$.
- Carol makes a guess $z$ about $x$.
- If $z = 0$, Carol takes $C = A$, else Carol takes $C = HA$.
- Carol measures $C$ to get the classical bit $k$.
- Carol sends the measured qubit $D$ to Bob.
- Bob processes $D$ as if he has received $A$ from Alice.

Later, Alice and Bob disclose $x$ and $y$. 
The BB84 Algorithm: Passive Eavesdropping

- Carol intercepts $A$.
- Carol makes a guess $z$ about $x$.
- If $z = 0$, Carol takes $C = A$, else Carol takes $C = HA$.
- Carol measures $C$ to get the classical bit $k$.
- Carol sends the measured qubit $D$ to Bob.
- Bob processes $D$ as if he has received $A$ from Alice.

Later, Alice and Bob disclose $x$ and $y$.

If $x \neq y$, the bits $i, j, k$ are discarded.
The BB84 Algorithm: Passive Eavesdropping

- Carol intercepts $A$.
- Carol makes a guess $z$ about $x$.
- If $z = 0$, Carol takes $C = A$, else Carol takes $C = HA$.
- Carol measures $C$ to get the classical bit $k$.
- Carol sends the measured qubit $D$ to Bob.
- Bob processes $D$ as if he has received $A$ from Alice.

Later, Alice and Bob disclose $x$ and $y$.
- If $x \neq y$, the bits $i, j, k$ are discarded.
- If $x = y$, Alice stores $i$, and Bob stores $j$. 
The BB84 Algorithm: Passive Eavesdropping

- Carol intercepts $A$.
- Carol makes a guess $z$ about $x$.
- If $z = 0$, Carol takes $C = A$, else Carol takes $C = HA$.
- Carol measures $C$ to get the classical bit $k$.
- Carol sends the measured qubit $D$ to Bob.
- Bob processes $D$ as if he has received $A$ from Alice.

- Later, Alice and Bob disclose $x$ and $y$.
- If $x \neq y$, the bits $i, j, k$ are discarded.
- If $x = y$, Alice stores $i$, and Bob stores $j$.
- Carol may have caused $i \neq j$ even when $x = y$. 
### The BB84 Algorithm: Eavesdropping Example

<table>
<thead>
<tr>
<th>Iter</th>
<th>i</th>
<th>x</th>
<th>A</th>
<th>z</th>
<th>C = H²A</th>
<th>k</th>
<th>D</th>
<th>y</th>
<th>B = HʸD</th>
<th>j</th>
</tr>
</thead>
</table>

- **Public-key Cryptography: Theory and Practice**
- **Abhijit Das**
### The BB84 Algorithm: Eavesdropping Example

<table>
<thead>
<tr>
<th>Iter</th>
<th>$i$</th>
<th>$x$</th>
<th>$A$</th>
<th>$z$</th>
<th>$C = H^2 A$</th>
<th>$k$</th>
<th>$D$</th>
<th>$y$</th>
<th>$B = H^y D$</th>
<th>$j$</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>$\frac{1}{\sqrt{2}}(</td>
<td>0\rangle +</td>
<td>1\rangle)$</td>
<td>1</td>
<td>$</td>
<td>0\rangle$</td>
<td>0</td>
<td>$</td>
</tr>
</tbody>
</table>
The BB84 Algorithm: Eavesdropping Example

<table>
<thead>
<tr>
<th>Iter</th>
<th>$i$</th>
<th>$x$</th>
<th>$A$</th>
<th>$z$</th>
<th>$C = H^2 A$</th>
<th>$k$</th>
<th>$D$</th>
<th>$y$</th>
<th>$B = H^y D$</th>
<th>$j$</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>$\frac{1}{\sqrt{2}}(</td>
<td>0\rangle +</td>
<td>1\rangle)$</td>
<td>1</td>
<td>$</td>
<td>0\rangle$</td>
<td>0</td>
<td>$</td>
</tr>
<tr>
<td>2</td>
<td>1</td>
<td>0</td>
<td>$</td>
<td>1\rangle$</td>
<td>0</td>
<td>$</td>
<td>1\rangle$</td>
<td>1</td>
<td>$</td>
<td>1\rangle$</td>
</tr>
</tbody>
</table>
The BB84 Algorithm: Eavesdropping Example

<table>
<thead>
<tr>
<th>Iter</th>
<th>$i$</th>
<th>$x$</th>
<th>$A$</th>
<th>$z$</th>
<th>$C = H^2A$</th>
<th>$k$</th>
<th>$D$</th>
<th>$y$</th>
<th>$B = H^yD$</th>
<th>$j$</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>$\frac{1}{\sqrt{2}}(</td>
<td>0\rangle +</td>
<td>1\rangle)$</td>
<td>1</td>
<td>$</td>
<td>0\rangle$</td>
<td>0</td>
<td>$</td>
</tr>
<tr>
<td>2</td>
<td>1</td>
<td>0</td>
<td>$</td>
<td>1\rangle$</td>
<td>0</td>
<td>$</td>
<td>1\rangle$</td>
<td>1</td>
<td>$</td>
<td>1\rangle$</td>
</tr>
<tr>
<td>3</td>
<td>0</td>
<td>1</td>
<td>$</td>
<td>1\rangle$</td>
<td>1</td>
<td>$\frac{1}{\sqrt{2}}(</td>
<td>0\rangle -</td>
<td>1\rangle)$</td>
<td>0</td>
<td>$</td>
</tr>
</tbody>
</table>
### The BB84 Algorithm: Eavesdropping Example

<table>
<thead>
<tr>
<th>Iter</th>
<th>$i$</th>
<th>$x$</th>
<th>$A$</th>
<th>$z$</th>
<th>$C = H^2 A$</th>
<th>$k$</th>
<th>$D$</th>
<th>$y$</th>
<th>$B = H^y D$</th>
<th>$j$</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>$\frac{1}{\sqrt{2}}(</td>
<td>0\rangle +</td>
<td>1\rangle)$</td>
<td>1</td>
<td>$</td>
<td>0\rangle$</td>
<td>0</td>
<td>$</td>
</tr>
<tr>
<td>2</td>
<td>1</td>
<td>0</td>
<td>$</td>
<td>1\rangle$</td>
<td>0</td>
<td>$</td>
<td>1\rangle$</td>
<td>1</td>
<td>$</td>
<td>1\rangle$</td>
</tr>
<tr>
<td>3</td>
<td>0</td>
<td>0</td>
<td>$</td>
<td>1\rangle$</td>
<td>1</td>
<td>$\frac{1}{\sqrt{2}}(</td>
<td>0\rangle -</td>
<td>1\rangle)$</td>
<td>0</td>
<td>$</td>
</tr>
<tr>
<td>4</td>
<td>0</td>
<td>1</td>
<td>$\frac{1}{\sqrt{2}}(</td>
<td>0\rangle +</td>
<td>1\rangle)$</td>
<td>0</td>
<td>$\frac{1}{\sqrt{2}}(</td>
<td>0\rangle +</td>
<td>1\rangle)$</td>
<td>0</td>
</tr>
</tbody>
</table>
## The BB84 Algorithm: Eavesdropping Example

<table>
<thead>
<tr>
<th>Iter</th>
<th>i</th>
<th>x</th>
<th>$A$</th>
<th>$z$</th>
<th>$C = H^2 A$</th>
<th>k</th>
<th>$D$</th>
<th>y</th>
<th>$B = H^y D$</th>
<th>$j$</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>$\frac{1}{\sqrt{2}}(</td>
<td>0\rangle +</td>
<td>1\rangle)$</td>
<td>1</td>
<td>$</td>
<td>0\rangle$</td>
<td>0</td>
<td>$</td>
</tr>
<tr>
<td>2</td>
<td>1</td>
<td>0</td>
<td>$</td>
<td>1\rangle$</td>
<td>0</td>
<td>$</td>
<td>1\rangle$</td>
<td>1</td>
<td>$</td>
<td>1\rangle$</td>
</tr>
<tr>
<td>3</td>
<td>1</td>
<td>0</td>
<td>$</td>
<td>1\rangle$</td>
<td>1</td>
<td>$\frac{1}{\sqrt{2}}(</td>
<td>0\rangle -</td>
<td>1\rangle)$</td>
<td>0</td>
<td>$</td>
</tr>
<tr>
<td>4</td>
<td>0</td>
<td>1</td>
<td>$\frac{1}{\sqrt{2}}(</td>
<td>0\rangle +</td>
<td>1\rangle)$</td>
<td>0</td>
<td>$\frac{1}{\sqrt{2}}(</td>
<td>0\rangle +</td>
<td>1\rangle)$</td>
<td>0</td>
</tr>
<tr>
<td>5</td>
<td>0</td>
<td>1</td>
<td>$\frac{1}{\sqrt{2}}(</td>
<td>0\rangle +</td>
<td>1\rangle)$</td>
<td>1</td>
<td>$</td>
<td>0\rangle$</td>
<td>0</td>
<td>$</td>
</tr>
</tbody>
</table>
**The BB84 Algorithm: Eavesdropping Example**

<table>
<thead>
<tr>
<th>Iter</th>
<th>i</th>
<th>x</th>
<th>A</th>
<th>z</th>
<th>C = H²A</th>
<th>k</th>
<th>D</th>
<th>y</th>
<th>B = H⁵D</th>
<th>j</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1/√2(</td>
<td>0⟩ +</td>
<td>1⟩)</td>
<td>1</td>
<td></td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td>1</td>
<td>0</td>
<td></td>
<td>0</td>
<td></td>
<td>1</td>
<td>1</td>
<td>0</td>
<td></td>
<td>1</td>
</tr>
<tr>
<td>3</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1/√2(</td>
<td>0⟩ −</td>
<td>1⟩)</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>4</td>
<td>0</td>
<td>1</td>
<td>1/√2(</td>
<td>0⟩ +</td>
<td>1⟩)</td>
<td>0</td>
<td>1/√2(</td>
<td>0⟩ +</td>
<td>1⟩)</td>
<td>0</td>
</tr>
<tr>
<td>5</td>
<td>0</td>
<td>1</td>
<td>1/√2(</td>
<td>0⟩ +</td>
<td>1⟩)</td>
<td>1</td>
<td></td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>6</td>
<td>1</td>
<td>1</td>
<td>1/√2(</td>
<td>0⟩ −</td>
<td>1⟩)</td>
<td>1</td>
<td></td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>
## The BB84 Algorithm: Eavesdropping Example

<table>
<thead>
<tr>
<th>Iter</th>
<th>i</th>
<th>x</th>
<th>A</th>
<th>z</th>
<th>C = H²A</th>
<th>k</th>
<th>D</th>
<th>y</th>
<th>B = H^yD</th>
<th>j</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1/√2(</td>
<td>0⟩ +</td>
<td>1⟩)</td>
<td>1</td>
<td></td>
<td>0⟩</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>1</td>
<td>0</td>
<td></td>
<td>1⟩</td>
<td>0</td>
<td></td>
<td>1⟩</td>
<td>1</td>
<td></td>
<td>1⟩</td>
</tr>
<tr>
<td>3</td>
<td>1</td>
<td>0</td>
<td></td>
<td>1⟩</td>
<td>1</td>
<td>1/√2(</td>
<td>0⟩ −</td>
<td>1⟩)</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>0</td>
<td>1</td>
<td>1/√2(</td>
<td>0⟩ +</td>
<td>1⟩)</td>
<td>0</td>
<td>1/√2(</td>
<td>0⟩ +</td>
<td>1⟩)</td>
<td>0</td>
</tr>
<tr>
<td>5</td>
<td>0</td>
<td>1</td>
<td>1/√2(</td>
<td>0⟩ +</td>
<td>1⟩)</td>
<td>1</td>
<td></td>
<td>0⟩</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>1</td>
<td>1</td>
<td>1/√2(</td>
<td>0⟩ −</td>
<td>1⟩)</td>
<td>1</td>
<td></td>
<td>1⟩</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>1</td>
<td>1</td>
<td>1/√2(</td>
<td>0⟩ −</td>
<td>1⟩)</td>
<td>0</td>
<td>1/√2(</td>
<td>0⟩ −</td>
<td>1⟩)</td>
<td>0</td>
</tr>
</tbody>
</table>
The BB84 Algorithm: Eavesdropping Example

<table>
<thead>
<tr>
<th>Iter</th>
<th>i</th>
<th>x</th>
<th>A</th>
<th>z</th>
<th>C = H²A</th>
<th>k</th>
<th>D</th>
<th>y</th>
<th>B = H⁷D</th>
<th>j</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>( \frac{1}{\sqrt{2}}(</td>
<td>0\rangle +</td>
<td>1\rangle) )</td>
<td>1</td>
<td>(</td>
<td>0\rangle )</td>
<td>0</td>
<td>(</td>
</tr>
<tr>
<td>2</td>
<td>1</td>
<td>0</td>
<td>(</td>
<td>1\rangle )</td>
<td>0</td>
<td>(</td>
<td>1\rangle )</td>
<td>1</td>
<td>(</td>
<td>1\rangle )</td>
</tr>
<tr>
<td>3</td>
<td>1</td>
<td>0</td>
<td>(</td>
<td>1\rangle )</td>
<td>1</td>
<td>( \frac{1}{\sqrt{2}}(</td>
<td>0\rangle -</td>
<td>1\rangle) )</td>
<td>0</td>
<td>(</td>
</tr>
<tr>
<td>4</td>
<td>0</td>
<td>1</td>
<td>( \frac{1}{\sqrt{2}}(</td>
<td>0\rangle +</td>
<td>1\rangle) )</td>
<td>0</td>
<td>( \frac{1}{\sqrt{2}}(</td>
<td>0\rangle +</td>
<td>1\rangle) )</td>
<td>0</td>
</tr>
<tr>
<td>5</td>
<td>0</td>
<td>1</td>
<td>( \frac{1}{\sqrt{2}}(</td>
<td>0\rangle +</td>
<td>1\rangle) )</td>
<td>1</td>
<td>(</td>
<td>0\rangle )</td>
<td>0</td>
<td>(</td>
</tr>
<tr>
<td>6</td>
<td>1</td>
<td>1</td>
<td>( \frac{1}{\sqrt{2}}(</td>
<td>0\rangle -</td>
<td>1\rangle) )</td>
<td>1</td>
<td>(</td>
<td>1\rangle )</td>
<td>1</td>
<td>(</td>
</tr>
<tr>
<td>7</td>
<td>1</td>
<td>1</td>
<td>( \frac{1}{\sqrt{2}}(</td>
<td>0\rangle -</td>
<td>1\rangle) )</td>
<td>0</td>
<td>( \frac{1}{\sqrt{2}}(</td>
<td>0\rangle -</td>
<td>1\rangle) )</td>
<td>0</td>
</tr>
<tr>
<td>8</td>
<td>1</td>
<td>0</td>
<td>(</td>
<td>1\rangle )</td>
<td>0</td>
<td>(</td>
<td>1\rangle )</td>
<td>1</td>
<td>(</td>
<td>1\rangle )</td>
</tr>
</tbody>
</table>
### The BB84 Algorithm: Eavesdropping Example

<table>
<thead>
<tr>
<th>Iter</th>
<th>$i$</th>
<th>$x$</th>
<th>$A$</th>
<th>$z$</th>
<th>$C = H^2 A$</th>
<th>$k$</th>
<th>$D$</th>
<th>$y$</th>
<th>$B = H^D$</th>
<th>$j$</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>$\frac{1}{\sqrt{2}}(</td>
<td>0\rangle +</td>
<td>1\rangle)$</td>
<td>1</td>
<td>$</td>
<td>0\rangle$</td>
<td>0</td>
<td>$</td>
</tr>
<tr>
<td>2</td>
<td>1</td>
<td>0</td>
<td>$</td>
<td>1\rangle$</td>
<td>0</td>
<td>$</td>
<td>1\rangle$</td>
<td>1</td>
<td>$</td>
<td>1\rangle$</td>
</tr>
<tr>
<td>3</td>
<td>1</td>
<td>0</td>
<td>$</td>
<td>1\rangle$</td>
<td>1</td>
<td>$\frac{1}{\sqrt{2}}(</td>
<td>0\rangle -</td>
<td>1\rangle)$</td>
<td>0</td>
<td>$</td>
</tr>
<tr>
<td>4</td>
<td>0</td>
<td>1</td>
<td>$\frac{1}{\sqrt{2}}(</td>
<td>0\rangle +</td>
<td>1\rangle)$</td>
<td>0</td>
<td>$\frac{1}{\sqrt{2}}(</td>
<td>0\rangle +</td>
<td>1\rangle)$</td>
<td>0</td>
</tr>
<tr>
<td>5</td>
<td>0</td>
<td>1</td>
<td>$\frac{1}{\sqrt{2}}(</td>
<td>0\rangle +</td>
<td>1\rangle)$</td>
<td>1</td>
<td>$</td>
<td>0\rangle$</td>
<td>0</td>
<td>$</td>
</tr>
<tr>
<td>6</td>
<td>1</td>
<td>1</td>
<td>$\frac{1}{\sqrt{2}}(</td>
<td>0\rangle -</td>
<td>1\rangle)$</td>
<td>1</td>
<td>$</td>
<td>1\rangle$</td>
<td>1</td>
<td>$</td>
</tr>
<tr>
<td>7</td>
<td>1</td>
<td>1</td>
<td>$\frac{1}{\sqrt{2}}(</td>
<td>0\rangle -</td>
<td>1\rangle)$</td>
<td>0</td>
<td>$\frac{1}{\sqrt{2}}(</td>
<td>0\rangle -</td>
<td>1\rangle)$</td>
<td>0</td>
</tr>
<tr>
<td>8</td>
<td>1</td>
<td>0</td>
<td>$</td>
<td>1\rangle$</td>
<td>0</td>
<td>$</td>
<td>1\rangle$</td>
<td>1</td>
<td>$</td>
<td>1\rangle$</td>
</tr>
<tr>
<td>9</td>
<td>1</td>
<td>1</td>
<td>$\frac{1}{\sqrt{2}}(</td>
<td>0\rangle -</td>
<td>1\rangle)$</td>
<td>0</td>
<td>$\frac{1}{\sqrt{2}}(</td>
<td>0\rangle -</td>
<td>1\rangle)$</td>
<td>0</td>
</tr>
</tbody>
</table>
### The BB84 Algorithm: Eavesdropping Example

<table>
<thead>
<tr>
<th>Iter</th>
<th>i</th>
<th>x</th>
<th>A</th>
<th>z</th>
<th>C = H^2 A</th>
<th>k</th>
<th>D</th>
<th>y</th>
<th>B = H^α D</th>
<th>j</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>( \frac{1}{\sqrt{2}} (</td>
<td>0\rangle +</td>
<td>1\rangle) )</td>
<td>1</td>
<td></td>
<td>0</td>
<td>0</td>
<td>1 ( \frac{1}{\sqrt{2}} (</td>
</tr>
<tr>
<td>2</td>
<td>1</td>
<td>0</td>
<td></td>
<td>0</td>
<td>(</td>
<td>0\rangle )</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1 (</td>
</tr>
<tr>
<td>3</td>
<td>1</td>
<td>0</td>
<td>(</td>
<td>1\rangle )</td>
<td>1</td>
<td>( \frac{1}{\sqrt{2}} (</td>
<td>0\rangle -</td>
<td>1\rangle) )</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>4</td>
<td>0</td>
<td>1</td>
<td>( \frac{1}{\sqrt{2}} (</td>
<td>0\rangle +</td>
<td>1\rangle) )</td>
<td>0</td>
<td>( \frac{1}{\sqrt{2}} (</td>
<td>0\rangle +</td>
<td>1\rangle) )</td>
<td>0</td>
</tr>
<tr>
<td>5</td>
<td>0</td>
<td>1</td>
<td>( \frac{1}{\sqrt{2}} (</td>
<td>0\rangle +</td>
<td>1\rangle) )</td>
<td>1</td>
<td></td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>6</td>
<td>1</td>
<td>1</td>
<td>( \frac{1}{\sqrt{2}} (</td>
<td>0\rangle -</td>
<td>1\rangle) )</td>
<td>1</td>
<td>(</td>
<td>1\rangle )</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>7</td>
<td>1</td>
<td>1</td>
<td>( \frac{1}{\sqrt{2}} (</td>
<td>0\rangle -</td>
<td>1\rangle) )</td>
<td>0</td>
<td>( \frac{1}{\sqrt{2}} (</td>
<td>0\rangle -</td>
<td>1\rangle) )</td>
<td>0</td>
</tr>
<tr>
<td>8</td>
<td>1</td>
<td>0</td>
<td>(</td>
<td>1\rangle )</td>
<td>0</td>
<td>(</td>
<td>1\rangle )</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>9</td>
<td>1</td>
<td>1</td>
<td>( \frac{1}{\sqrt{2}} (</td>
<td>0\rangle -</td>
<td>1\rangle) )</td>
<td>0</td>
<td>( \frac{1}{\sqrt{2}} (</td>
<td>0\rangle -</td>
<td>1\rangle) )</td>
<td>0</td>
</tr>
<tr>
<td>10</td>
<td>0</td>
<td>1</td>
<td>( \frac{1}{\sqrt{2}} (</td>
<td>0\rangle +</td>
<td>1\rangle) )</td>
<td>0</td>
<td>( \frac{1}{\sqrt{2}} (</td>
<td>0\rangle +</td>
<td>1\rangle) )</td>
<td>1</td>
</tr>
</tbody>
</table>
### The BB84 Algorithm: Eavesdropping Example

<table>
<thead>
<tr>
<th>Iter</th>
<th>$i$</th>
<th>$x$</th>
<th>$A$</th>
<th>$z$</th>
<th>$C = H^2 A$</th>
<th>$k$</th>
<th>$D$</th>
<th>$y$</th>
<th>$B = H^y D$</th>
<th>$j$</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>$\frac{1}{\sqrt{2}}(</td>
<td>0\rangle +</td>
<td>1\rangle)$</td>
<td>1</td>
<td>$</td>
<td>0\rangle$</td>
<td>0</td>
<td>$</td>
</tr>
<tr>
<td>2</td>
<td>1</td>
<td>0</td>
<td>$</td>
<td>1\rangle$</td>
<td>0</td>
<td>$</td>
<td>1\rangle$</td>
<td>1</td>
<td>$</td>
<td>1\rangle$</td>
</tr>
<tr>
<td>3</td>
<td>0</td>
<td>0</td>
<td>$</td>
<td>1\rangle$</td>
<td>1</td>
<td>$\frac{1}{\sqrt{2}}(</td>
<td>0\rangle -</td>
<td>1\rangle)$</td>
<td>0</td>
<td>$</td>
</tr>
<tr>
<td>4</td>
<td>0</td>
<td>1</td>
<td>$\frac{1}{\sqrt{2}}(</td>
<td>0\rangle +</td>
<td>1\rangle)$</td>
<td>0</td>
<td>$\frac{1}{\sqrt{2}}(</td>
<td>0\rangle +</td>
<td>1\rangle)$</td>
<td>0</td>
</tr>
<tr>
<td>5</td>
<td>0</td>
<td>1</td>
<td>$\frac{1}{\sqrt{2}}(</td>
<td>0\rangle +</td>
<td>1\rangle)$</td>
<td>1</td>
<td>$</td>
<td>0\rangle$</td>
<td>0</td>
<td>$</td>
</tr>
<tr>
<td>6</td>
<td>1</td>
<td>1</td>
<td>$\frac{1}{\sqrt{2}}(</td>
<td>0\rangle -</td>
<td>1\rangle)$</td>
<td>1</td>
<td>$</td>
<td>1\rangle$</td>
<td>1</td>
<td>$</td>
</tr>
<tr>
<td>7</td>
<td>1</td>
<td>1</td>
<td>$\frac{1}{\sqrt{2}}(</td>
<td>0\rangle -</td>
<td>1\rangle)$</td>
<td>0</td>
<td>$\frac{1}{\sqrt{2}}(</td>
<td>0\rangle -</td>
<td>1\rangle)$</td>
<td>0</td>
</tr>
<tr>
<td>8</td>
<td>1</td>
<td>0</td>
<td>$</td>
<td>1\rangle$</td>
<td>0</td>
<td>$</td>
<td>1\rangle$</td>
<td>1</td>
<td>$</td>
<td>1\rangle$</td>
</tr>
<tr>
<td>9</td>
<td>1</td>
<td>1</td>
<td>$\frac{1}{\sqrt{2}}(</td>
<td>0\rangle -</td>
<td>1\rangle)$</td>
<td>0</td>
<td>$\frac{1}{\sqrt{2}}(</td>
<td>0\rangle -</td>
<td>1\rangle)$</td>
<td>0</td>
</tr>
<tr>
<td>10</td>
<td>0</td>
<td>1</td>
<td>$\frac{1}{\sqrt{2}}(</td>
<td>0\rangle +</td>
<td>1\rangle)$</td>
<td>0</td>
<td>$\frac{1}{\sqrt{2}}(</td>
<td>0\rangle +</td>
<td>1\rangle)$</td>
<td>1</td>
</tr>
</tbody>
</table>

$i$ and $j$ differ in five positions.
The BB84 Algorithm: Security
The BB84 Algorithm: Security

• It is impossible to copy a qubit.
The BB84 Algorithm: Security

- It is impossible to copy a qubit.
- It is impossible to restore a qubit to a pre-measurement state.
The BB84 Algorithm: Security

- It is impossible to copy a qubit.
- It is impossible to restore a qubit to a pre-measurement state.
- The more Carol eavesdrops, the more she forces $i \neq j$. 
The BB84 Algorithm: Security

- It is impossible to copy a qubit.
- It is impossible to restore a qubit to a pre-measurement state.
- The more Carol eavesdrops, the more she forces $i \neq j$.
- Carol’s presence can be detected by Alice and Bob.
The BB84 Algorithm: Security

- It is impossible to copy a qubit.
- It is impossible to restore a qubit to a pre-measurement state.
- The more Carol eavesdrops, the more she forces $i \neq j$.
- Carol’s presence can be detected by Alice and Bob.
- There is no need to reveal the shared secret.
The BB84 Algorithm: Security

- It is impossible to copy a qubit.
- It is impossible to restore a qubit to a pre-measurement state.
- The more Carol eavesdrops, the more she forces $i \neq j$.
- Carol’s presence can be detected by Alice and Bob.
- There is no need to reveal the shared secret.

  - Alice and Bob may transmit parity check bits at regular intervals.
The BB84 Algorithm: Security

- It is impossible to copy a qubit.
- It is impossible to restore a qubit to a pre-measurement state.
- The more Carol eavesdrops, the more she forces $i \neq j$.
- Carol’s presence can be detected by Alice and Bob.
- There is no need to reveal the shared secret.

  - Alice and Bob may transmit parity check bits at regular intervals.
  - Alternatively, Alice and Bob may exchange plaintext-ciphertext pairs based on their shared keys.
The BB84 Algorithm: Security

- It is impossible to copy a qubit.
- It is impossible to restore a qubit to a pre-measurement state.
- The more Carol eavesdrops, the more she forces $i \neq j$.
- Carol’s presence can be detected by Alice and Bob.
- There is no need to reveal the shared secret.
  - Alice and Bob may transmit parity check bits at regular intervals.
  - Alternatively, Alice and Bob may exchange plaintext-ciphertext pairs based on their shared keys.
- If eavesdropping is detected, the key exchange session is discarded.
The BB84 Algorithm: Practical Implementation
The BB84 Algorithm: Practical Implementation

- Polarization of photons can be used.

<table>
<thead>
<tr>
<th>Polarization angle</th>
<th>Qubit value</th>
</tr>
</thead>
<tbody>
<tr>
<td>0°</td>
<td></td>
</tr>
<tr>
<td>45°</td>
<td>( \frac{1}{\sqrt{2}}(</td>
</tr>
<tr>
<td>90°</td>
<td></td>
</tr>
<tr>
<td>135°</td>
<td>( \frac{1}{\sqrt{2}}(</td>
</tr>
</tbody>
</table>
The BB84 Algorithm: Practical Implementation

- Polarization of photons can be used.

<table>
<thead>
<tr>
<th>Polarization angle</th>
<th>Qubit value</th>
</tr>
</thead>
<tbody>
<tr>
<td>$0^0$</td>
<td>$</td>
</tr>
<tr>
<td>$45^0$</td>
<td>$\frac{1}{\sqrt{2}}(</td>
</tr>
<tr>
<td>$90^0$</td>
<td>$</td>
</tr>
<tr>
<td>$135^0$</td>
<td>$\frac{1}{\sqrt{2}}(</td>
</tr>
</tbody>
</table>

- A $45^0$ filter is used to implement the Hadamard transform $H$. 

Public-key Cryptography: Theory and Practice

Abhijit Das
The BB84 Algorithm: Practical Implementation

- Polarization of photons can be used.

<table>
<thead>
<tr>
<th>Polarization angle</th>
<th>Qubit value</th>
</tr>
</thead>
<tbody>
<tr>
<td>0°</td>
<td>$</td>
</tr>
<tr>
<td>45°</td>
<td>$\frac{1}{\sqrt{2}}(</td>
</tr>
<tr>
<td>90°</td>
<td>$</td>
</tr>
<tr>
<td>135°</td>
<td>$\frac{1}{\sqrt{2}}(</td>
</tr>
</tbody>
</table>

- A 45° filter is used to implement the Hadamard transform $H$.
- Bennett and Brassard did the first implementation in the T. J. Watson Research Center.
Polarization of photons can be used.

<table>
<thead>
<tr>
<th>Polarization angle</th>
<th>Qubit value</th>
</tr>
</thead>
<tbody>
<tr>
<td>$0^0$</td>
<td>$</td>
</tr>
<tr>
<td>$45^0$</td>
<td>$\frac{1}{\sqrt{2}}(</td>
</tr>
<tr>
<td>$90^0$</td>
<td>$</td>
</tr>
<tr>
<td>$135^0$</td>
<td>$\frac{1}{\sqrt{2}}(</td>
</tr>
</tbody>
</table>

A $45^0$ filter is used to implement the Hadamard transform $H$.

Bennett and Brassard did the first implementation in the T. J. Watson Research Center.

They used a quantum channel of length 32 cm.
The BB84 Algorithm: Practical Implementation

- Polarization of photons can be used.
  
<table>
<thead>
<tr>
<th>Polarization angle</th>
<th>Qubit value</th>
</tr>
</thead>
<tbody>
<tr>
<td>0°</td>
<td>$</td>
</tr>
<tr>
<td>45°</td>
<td>$\frac{1}{\sqrt{2}}(</td>
</tr>
<tr>
<td>90°</td>
<td>$</td>
</tr>
<tr>
<td>135°</td>
<td>$\frac{1}{\sqrt{2}}(</td>
</tr>
</tbody>
</table>

- A 45° filter is used to implement the Hadamard transform $H$.
- Bennett and Brassard did the first implementation in the T. J. Watson Research Center.
- They used a quantum channel of length 32 cm.
- Current record: 148.7 km (Los Alamos/NIST).
Shor’s Algorithm: Introduction
Let $m$ be an odd integer that we want to factor.
Shor’s Algorithm: Introduction

- Let $m$ be an odd integer that we want to factor.
- Choose $a \in \mathbb{Z}_m^*$. 
Let $m$ be an odd integer that we want to factor.
Choose $a \in \mathbb{Z}_m^*$. 
Let $r$ be the multiplicative order of $a$ modulo $m$. 
Shor’s Algorithm: Introduction

- Let $m$ be an odd integer that we want to factor.
- Choose $a \in \mathbb{Z}_m^*$.
- Let $r$ be the multiplicative order of $a$ modulo $m$.
- Choose $n \in \mathbb{N}$ with $N = 2^n \geq m^2 > r^2$. 
Shor’s Algorithm: Introduction

- Let \( m \) be an odd integer that we want to factor.
- Choose \( a \in \mathbb{Z}_m^* \).
- Let \( r \) be the multiplicative order of \( a \) modulo \( m \).
- Choose \( n \in \mathbb{N} \) with \( N = 2^n \geq m^2 > r^2 \).
- The function \( f : \mathbb{Z} \rightarrow \mathbb{Z}_N \) taking \( x \rightarrow a^x \pmod{m} \) is periodic of least period \( r \).
Shor’s Algorithm: Introduction

- Let $m$ be an odd integer that we want to factor.
- Choose $a \in \mathbb{Z}_m^*$.
- Let $r$ be the multiplicative order of $a$ modulo $m$.
- Choose $n \in \mathbb{N}$ with $N = 2^n \geq m^2 > r^2$.
- The function $f : \mathbb{Z} \rightarrow \mathbb{Z}_N$ taking $x \mapsto a^x \pmod{m}$ is periodic of least period $r$.
- Shor’s algorithm computes $r$. 
Let $m$ be an odd integer that we want to factor.
Choose $a \in \mathbb{Z}_m^*$.
Let $r$ be the multiplicative order of $a$ modulo $m$.
Choose $n \in \mathbb{N}$ with $N = 2^n \geq m^2 > r^2$.
The function $f : \mathbb{Z} \to \mathbb{Z}_N$ taking $x \mapsto a^x \pmod{m}$ is periodic of least period $r$.
Shor’s algorithm computes $r$.
If $r$ is even, $(a^{r/2} - 1)(a^{r/2} + 1) \equiv 0 \pmod{m}$.
Shor’s Algorithm: Introduction

- Let $m$ be an odd integer that we want to factor.
- Choose $a \in \mathbb{Z}_m^*$.
- Let $r$ be the multiplicative order of $a$ modulo $m$.
- Choose $n \in \mathbb{N}$ with $N = 2^n \geq m^2 > r^2$.
- The function $f : \mathbb{Z} \to \mathbb{Z}_N$ taking $x \mapsto a^x \pmod{m}$ is periodic of least period $r$.
- Shor’s algorithm computes $r$.
- If $r$ is even, $(a^{r/2} - 1)(a^{r/2} + 1) \equiv 0 \pmod{m}$.
- With probability at least $1/2$, we have $a^{r/2} + 1 \not\equiv 0 \pmod{m}$.
Shor’s Algorithm: Introduction

- Let $m$ be an odd integer that we want to factor.
- Choose $a \in \mathbb{Z}_m^*$.
- Let $r$ be the multiplicative order of $a$ modulo $m$.
- Choose $n \in \mathbb{N}$ with $N = 2^n \geq m^2 > r^2$.
- The function $f : \mathbb{Z} \to \mathbb{Z}_N$ taking $x \mapsto a^x \pmod{m}$ is periodic of least period $r$.
- Shor’s algorithm computes $r$.
- If $r$ is even, $(a^{r/2} - 1)(a^{r/2} + 1) \equiv 0 \pmod{m}$.
- With probability at least $1/2$, we have $a^{r/2} + 1 \not\equiv 0 \pmod{m}$.
- If so, $\gcd(a^{r/2} + 1, m)$ is a non-trivial factor of $m$. 
Shor’s Algorithm: Introduction

- Let $m$ be an odd integer that we want to factor.
- Choose $a \in \mathbb{Z}_m^*$.
- Let $r$ be the multiplicative order of $a$ modulo $m$.
- Choose $n \in \mathbb{N}$ with $N = 2^n \geq m^2 > r^2$.
- The function $f : \mathbb{Z} \to \mathbb{Z}_N$ taking $x \mapsto a^x \pmod{m}$ is periodic of least period $r$.
- Shor’s algorithm computes $r$.
- If $r$ is even, $(a^{r/2} - 1)(a^{r/2} + 1) \equiv 0 \pmod{m}$.
- With probability at least $1/2$, we have $a^{r/2} + 1 \not\equiv 0 \pmod{m}$.
- If so, $\gcd(a^{r/2} + 1, m)$ is a non-trivial factor of $m$.
- If not (or if $r$ is odd), repeat with another $a$. 
Shor's Algorithm: A Classical Approach
Shor’s Algorithm: A Classical Approach

Evaluate $f(x)$ for many values of $x$. 
Shor's Algorithm: A Classical Approach

- Evaluate $f(x)$ for many values of $x$.
- Once we find $x$ and $y$ with $f(x) = f(y)$, we have $r \mid (x - y)$. 
Shor’s Algorithm: A Classical Approach

- Evaluate $f(x)$ for many values of $x$.
- Once we find $x$ and $y$ with $f(x) = f(y)$, we have $r | (x - y)$.
- $r$ can be determined by taking the gcd of a few such values of $x - y$. 

Public-key Cryptography: Theory and Practice

Abhijit Das
Shor’s Algorithm: A Classical Approach

- Evaluate \( f(x) \) for many values of \( x \).
- Once we find \( x \) and \( y \) with \( f(x) = f(y) \), we have \( r \mid (x - y) \).
- \( r \) can be determined by taking the gcd of a few such values of \( x - y \).
- By the birthday paradox, we need \( O(\sqrt{r}) \) evaluations of \( f \) to obtain a collision \( f(x) = f(y) \).
Shor’s Algorithm: A Classical Approach

- Evaluate $f(x)$ for many values of $x$.
- Once we find $x$ and $y$ with $f(x) = f(y)$, we have $r \mid (x - y)$.
- $r$ can be determined by taking the gcd of a few such values of $x - y$.
- By the birthday paradox, we need $O(\sqrt{r})$ evaluations of $f$ to obtain a collision $f(x) = f(y)$.
- But $r$ can be large, like $r \approx m$. 

Public-key Cryptography: Theory and Practice  
Abhijit Das
Shor’s Algorithm: A Classical Approach

- Evaluate $f(x)$ for many values of $x$.
- Once we find $x$ and $y$ with $f(x) = f(y)$, we have $r \mid (x - y)$.
- $r$ can be determined by taking the gcd of a few such values of $x - y$.
- By the birthday paradox, we need $O(\sqrt{r})$ evaluations of $f$ to obtain a collision $f(x) = f(y)$.
- But $r$ can be large, like $r \approx m$.
- The classical algorithm may take exponential time (in $\log m$).
Shor’s Algorithm: A Classical Approach

- Evaluate $f(x)$ for many values of $x$.
- Once we find $x$ and $y$ with $f(x) = f(y)$, we have $r \mid (x - y)$.
- $r$ can be determined by taking the gcd of a few such values of $x - y$.
- By the birthday paradox, we need $O(\sqrt{r})$ evaluations of $f$ to obtain a collision $f(x) = f(y)$.
- But $r$ can be large, like $r \approx m$.
- The classical algorithm may take exponential time (in $\log m$).

- Shor’s algorithm computes $r$ with high probability by making only a single evaluation of $f$. 
Shor’s Algorithm: Preparation
Shor's Algorithm: Preparation

- Use a $2^n$-bit quantum register $R$. 
Shor’s Algorithm: Preparation

- Use a $2n$-bit quantum register $R$.
- Initialize $R$ to $|0\rangle_n |0\rangle_n$. 
Shor’s Algorithm: Preparation

- Use a $2n$-bit quantum register $R$.
- Initialize $R$ to $|0\rangle_n|0\rangle_n$.
- Apply the Hadamard transform to the left $n$ bits to obtain

$$
\left(H^{(n)} \otimes I^{(n)}\right) |0\rangle_n |0\rangle_n = \frac{1}{\sqrt{N}} \sum_{x=0}^{N-1} |x\rangle_n |0\rangle_n.
$$
Shor’s Algorithm: Preparation

- Use a $2n$-bit quantum register $R$.
- Initialize $R$ to $|0\rangle_n|0\rangle_n$.
- Apply the Hadamard transform to the left $n$ bits to obtain
  
  \[
  \left( H^{(n)} \otimes I^{(n)} \right) |0\rangle_n|0\rangle_n = \frac{1}{\sqrt{N}} \sum_{x=0}^{N-1} |x\rangle_n|0\rangle_n.
  \]
- Apply $f$ to change the state $|x\rangle_n|y\rangle_n$ to $|x\rangle_n|f(x) \oplus y\rangle_n$. 
Shor’s Algorithm: Preparation

- Use a $2n$-bit quantum register $R$.
- Initialize $R$ to $|0\rangle_n|0\rangle_n$.
- Apply the Hadamard transform to the left $n$ bits to obtain the state
  
  $$\left(H^{(n)} \otimes I^{(n)}\right)|0\rangle_n|0\rangle_n = \frac{1}{\sqrt{N}} \sum_{x=0}^{N-1} |x\rangle_n|0\rangle_n.$$ 

- Apply $f$ to change the state $|x\rangle_n|y\rangle_n$ to $|x\rangle_n|f(x) \oplus y\rangle_n$.
- $R$ switches to the state
  
  $$\frac{1}{\sqrt{N}} \sum_{x=0}^{N-1} |x\rangle_n|f(x)\rangle_n.$$
Shor’s Algorithm: Preparation

- Use a $2n$-bit quantum register $R$.
- Initialize $R$ to $|0\rangle_n|0\rangle_n$.
- Apply the Hadamard transform to the left $n$ bits to obtain
  
  $$
  \left( H^{(n)} \otimes I^{(n)} \right) |0\rangle_n|0\rangle_n = \frac{1}{\sqrt{N}} \sum_{x=0}^{N-1} |x\rangle_n|0\rangle_n.
  $$

- Apply $f$ to change the state $|x\rangle_n|y\rangle_n$ to $|x\rangle_n|f(x) \oplus y\rangle_n$.

- $R$ switches to the state $\frac{1}{\sqrt{N}} \sum_{x=0}^{N-1} |x\rangle_n|f(x)\rangle_n$.

- Evaluate the right $n$ bits. We get $f(x_0) \in \{0, 1, 2, \ldots, N - 1\}$ for some $x_0 \in \{0, 1, 2, \ldots, r - 1\}$. 
Shor’s Algorithm: Preparation

- Use a $2n$-bit quantum register $R$.
- Initialize $R$ to $|0\rangle_n|0\rangle_n$.
- Apply the Hadamard transform to the left $n$ bits to obtain
  \[
  \left( H^{(n)} \otimes I^{(n)} \right) |0\rangle_n|0\rangle_n = \frac{1}{\sqrt{N}} \sum_{x=0}^{N-1} |x\rangle_n|0\rangle_n.
  \]
- Apply $f$ to change the state $|x\rangle_n|y\rangle_n$ to $|x\rangle_n|f(x) \oplus y\rangle_n$.
- $R$ switches to the state $\frac{1}{\sqrt{N}} \sum_{x=0}^{N-1} |x\rangle_n|f(x)\rangle_n$.
- Evaluate the right $n$ bits. We get $f(x_0) \in \{0, 1, 2, \ldots, N-1\}$ for some $x_0 \in \{0, 1, 2, \ldots, r-1\}$.
- $R$ collapses to the state $\frac{1}{\sqrt{M}} \sum_{j=0}^{M-1} |x_0 + jr\rangle_n$, where $x_0 + (M - 1)r < N \leq x_0 + Mr$ (by the generalized Born rule).
Shor’s Algorithm: A Nice State, but . . .
Shor’s Algorithm: A Nice State, but . . .

- Suppose we are allowed to make copies of this state and measure these copies.
Suppose we are allowed to make copies of this state and measure these copies.

With high probability, we would get $x_0 + j\pi$ for different values of $j$. 

Shor’s Algorithm: A Nice State, but . . .
Shor’s Algorithm: A Nice State, but . . .

- Suppose we are allowed to make copies of this state and measure these copies.
- With high probability, we would get $x_0 + jr$ for different values of $j$.
- $r$ could be computed from these $x_0 + jr$ values.
Shor’s Algorithm: A Nice State, but . . .

- Suppose we are allowed to make copies of this state and measure these copies.
- With high probability, we would get $x_0 + jr$ for different values of $j$.
- $r$ could be computed from these $x_0 + jr$ values.
- This is impossible by the no-cloning theorem.
Shor’s Algorithm: A Nice State, but . . .

- Suppose we are allowed to make copies of this state and measure these copies.
- With high probability, we would get $x_0 + jr$ for different values of $j$.
- $r$ could be computed from these $x_0 + jr$ values.
- This is impossible by the no-cloning theorem.
- If we repeat the preparation steps afresh, $R$ gets the state
  \[
  \frac{1}{\sqrt{M}} \sum_{j=0}^{M'-1} |x_1 + jr\rangle_n \text{ in the left } n \text{ bits.}
  \]
Suppose we are allowed to make copies of this state and measure these copies.

With high probability, we would get $x_0 + jr$ for different values of $j$.

$r$ could be computed from these $x_0 + jr$ values.

This is impossible by the no-cloning theorem.

If we repeat the preparation steps afresh, $R$ gets the state

$$\frac{1}{\sqrt{M}} \sum_{j=0}^{M'-1} |x_1 + jr\rangle_n$$

in the left $n$ bits.

Now, measurement gives $x_1 + jr$. 
Shor’s Algorithm: A Nice State, but . . .

- Suppose we are allowed to make copies of this state and measure these copies.
- With high probability, we would get $x_0 + jr$ for different values of $j$.
- $r$ could be computed from these $x_0 + jr$ values.
- This is impossible by the no-cloning theorem.
- If we repeat the preparation steps afresh, $R$ gets the state
  \[
  \frac{1}{\sqrt{M}} \sum_{j=0}^{M'-1} |x_1 + jr\rangle_n \]
  in the left $n$ bits.
- Now, measurement gives $x_1 + jr$.
- With high probability, $x_0 \neq x_1$. 

Public-key Cryptography: Theory and Practice

Abhijit Das
Shor’s Algorithm: A Nice State, but . . .

- Suppose we are allowed to make copies of this state and measure these copies.
- With high probability, we would get $x_0 + jr$ for different values of $j$.
- $r$ could be computed from these $x_0 + jr$ values.
- This is impossible by the no-cloning theorem.
- If we repeat the preparation steps afresh, $R$ gets the state
  \[
  \frac{1}{\sqrt{M}} \sum_{j=0}^{M'-1} |x_1 + jr\rangle_n \text{ in the left } n \text{ bits.}
  \]
- Now, measurement gives $x_1 + jr$.
- With high probability, $x_0 \neq x_1$.
- Having a collision $x_u = x_v$ is governed by the birthday paradox, and the algorithm becomes exponential again.
Shor’s Algorithm: Fourier Transform, the Rescuer
Shor’s Algorithm: Fourier Transform, the Rescuer

- Use $n$-bit Fourier transform $F : |x\rangle_n \leftrightarrow \frac{1}{\sqrt{N}} \sum_{y=0}^{N-1} e^{2\pi i xy/N} |y\rangle_n$. 

\[
F : |x\rangle_n \leftrightarrow \frac{1}{\sqrt{N}} \sum_{y=0}^{N-1} e^{2\pi i xy/N} |y\rangle_n.
\]
Shor’s Algorithm: Fourier Transform, the Rescuer

- Use $n$-bit Fourier transform $F : |x\rangle_n \mapsto \frac{1}{\sqrt{N}} \sum_{y=0}^{N-1} e^{2\pi i xy/N} |y\rangle_n$.

- Application of $F$ on the left $n$ bits of $R$ available from the preparation stage gives the state

$$F \frac{1}{\sqrt{M}} \sum_{j=0}^{M-1} |x_0 + jr\rangle_n$$
Shor’s Algorithm: Fourier Transform, the Rescuer

- Use $n$-bit Fourier transform $F : |x\rangle_n \mapsto \frac{1}{\sqrt{N}} \sum_{y=0}^{N-1} e^{2\pi i x y/N} |y\rangle_n$.

- Application of $F$ on the left $n$ bits of $R$ available from the preparation stage gives the state

$$F \frac{1}{\sqrt{M}} \sum_{j=0}^{M-1} |x_0 + jr\rangle_n$$

$$= \frac{1}{\sqrt{N}} \sum_{y=0}^{N-1} \left( \frac{1}{\sqrt{M}} \sum_{j=0}^{M-1} e^{2\pi i (x_0+jr)y/N} |y\rangle_n \right)$$
Shor’s Algorithm: Fourier Transform, the Rescuer

- Use $n$-bit Fourier transform $F : |x⟩_n \mapsto \frac{1}{\sqrt{N}} \sum_{y=0}^{N-1} e^{2\pi i xy/N} |y⟩_n$.

- Application of $F$ on the left $n$ bits of $R$ available from the preparation stage gives the state

$$F \frac{1}{\sqrt{M}} \sum_{j=0}^{M-1} |x_0 + jr⟩_n$$

$$= \frac{1}{\sqrt{N}} \sum_{y=0}^{N-1} \left( \frac{1}{\sqrt{M}} \sum_{j=0}^{M-1} e^{2\pi i (x_0+jr)y/N} |y⟩_n \right)$$

$$= \frac{1}{\sqrt{NM}} \sum_{y=0}^{N-1} \left( e^{2\pi i x_0 y/N} \sum_{j=0}^{M-1} e^{2\pi i jry/N} \right) |y⟩_n.$$
Shor’s Algorithm: Final Steps
Shor’s Algorithm: Final Steps

- Measure the left $n$ bits of $R$ to get $y \in \{0, 1, 2, \ldots, N - 1\}$

with probability $p_y := \frac{1}{NM} \left| \sum_{j=0}^{M-1} e^{\frac{2\pi i j ry}{N}} \right|^2$. 
Shor’s Algorithm: Final Steps

- Measure the left $n$ bits of $R$ to get $y \in \{0, 1, 2, \ldots, N - 1\}$

  with probability
  
  $$p_y := \frac{1}{NM} \left| \sum_{j=0}^{M-1} e^{2\pi i j r y / N} \right|^2.$$

- $F$ changed the state from a uniform superposition to a state with higher probabilities for useful values.
Shor’s Algorithm: Final Steps

- Measure the left $n$ bits of $R$ to get $y \in \{0, 1, 2, \ldots, N - 1\}$ with probability $p_y := \frac{1}{NM} \left| \sum_{j=0}^{M-1} e^{2\pi i jry/N} \right|^2$.

- $F$ changed the state from a uniform superposition to a state with higher probabilities for useful values.

- A measurement $y$ is useful if its value is within $\pm \frac{1}{2}$ of an integral multiple of $N/r$. 
Shor’s Algorithm: Final Steps

- Measure the left $n$ bits of $R$ to get $y \in \{0, 1, 2, \ldots, N - 1\}$ with probability $p_y := \frac{1}{NM} \left| \sum_{j=0}^{M-1} e^{2\pi i jy/N} \right|^2$.

- $F$ changed the state from a uniform superposition to a state with higher probabilities for useful values.

- A measurement $y$ is useful if its value is within $\pm \frac{1}{2}$ of an integral multiple of $N/r$.

- The probability that we measure a useful $y$ is at least $\frac{4}{\pi^2} = 0.40528 \ldots$. 

...
Shor’s Algorithm: Final Steps

- Measure the left $n$ bits of $R$ to get $y \in \{0, 1, 2, \ldots, N - 1\}$ with probability
  \[ p_y := \frac{1}{NM} \left| \sum_{j=0}^{M-1} e^{2\pi i jr/N} \right|^2. \]

- $F$ changed the state from a uniform superposition to a state with higher probabilities for useful values.

- A measurement $y$ is useful if its value is within $\pm \frac{1}{2}$ of an integral multiple of $N/r$.

- The probability that we measure a useful $y$ is at least
  \[ \frac{4}{\pi^2} = 0.40528 \ldots. \]

- If the measured $y$ is useful, we run a classical algorithm (based upon continued fractions) to obtain a factor of $r$. 