Quantum computing for cryptographers III

Simon's algorithm

Tanja Lange
idea and design by Daniel J. Bernstein

Eindhoven University of Technology

SAC – Post-quantum cryptography

Simon's algorithm

Assumptions:

- Function $f : \mathbf{F}_2^n \to \{0,1\}^n$.
- Given any $u \in \mathbf{F}_2^n$.
  can efficiently compute $f(u)$.
- Nonzero $s \in \mathbf{F}_2^n$.
- $f(u) = f(u+s)$ for all u.
- f has no other collisions.

Goal: Figure out s.

Simon's algorithm

Assumptions:

- Function $f : \mathbf{F}_2^n \to \{0,1\}^n$.
- Given any $u \in \mathbf{F}_2^n$.
  can efficiently compute $f(u)$.
- Nonzero $s \in \mathbf{F}_2^n$.
- $f(u) = f(u+s)$ for all u.
- f has no other collisions.

Goal: Figure out s.

Traditional algorithm to find s:
compute f for many inputs,
hope to find collision.

Simon's algorithm

Assumptions:

- Function $f : \mathbf{F}_2^n \to \{0,1\}^n$.
- Given any $u \in \mathbf{F}_2^n$.
  can efficiently compute $f(u)$.
- Nonzero $s \in \mathbf{F}_2^n$.
- $f(u) = f(u+s)$ for all u.
- f has no other collisions.

Goal: Figure out s.

Traditional algorithm to find s:
compute f for many inputs,
hope to find collision.

Simon's algorithm finds s with
$\approx n$ reversible computations of f.

Simon's algorithm

Assumptions:

- Function $f : \mathbf{F}_2^n \to \{0,1\}^n$.
- Given any $u \in \mathbf{F}_2^n$. can efficiently compute $f(u)$.
- Nonzero $s \in \mathbf{F}_2^n$.
- $f(u) = f(u+s)$ for all u.
- f has no other collisions.

Goal: Figure out s.

Traditional algorithm to find s: compute f for many inputs, hope to find collision.

Simon's algorithm finds s with $\approx n$ reversible computations of f.

Step 1. Set up pure zero state:

1, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0.

Simon's algorithm

Assumptions:

- Function $f : \mathbf{F}_2^n \to \{0,1\}^n$.
- Given any $u \in \mathbf{F}_2^n$.
  can efficiently compute $f(u)$.
- Nonzero $s \in \mathbf{F}_2^n$.
- $f(u) = f(u+s)$ for all u.
- f has no other collisions.

Goal: Figure out s.

Traditional algorithm to find s:
compute f for many inputs,
hope to find collision.

Simon's algorithm finds s with
$\approx n$ reversible computations of f.

Step 2. Hadamard$_0$:

1, 1, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0.

Simon's algorithm

Assumptions:

- Function $f : \mathbf{F}_2^n \to \{0,1\}^n$.
- Given any $u \in \mathbf{F}_2^n$. can efficiently compute $f(u)$.
- Nonzero $s \in \mathbf{F}_2^n$.
- $f(u) = f(u+s)$ for all u.
- f has no other collisions.

Goal: Figure out s.

Traditional algorithm to find s: compute f for many inputs, hope to find collision.

Simon's algorithm finds s with $\approx n$ reversible computations of f.

Step 3. Hadamard$_1$:

1, 1, 1, 1, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0.

Simon's algorithm

Assumptions:

- Function $f : \mathbf{F}_2^n \to \{0,1\}^n$.
- Given any $u \in \mathbf{F}_2^n$.
  can efficiently compute $f(u)$.
- Nonzero $s \in \mathbf{F}_2^n$.
- $f(u) = f(u+s)$ for all u.
- f has no other collisions.

Goal: Figure out s.

Traditional algorithm to find s: compute f for many inputs, hope to find collision.

Simon's algorithm finds s with $\approx n$ reversible computations of f.

Step 4. Hadamard$_2$:
1, 1, 1, 1, 1, 1, 1, 1,
0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0.

Each column is a parallel universe.

Simon's algorithm

Assumptions:

- Function $f : \mathbf{F}_2^n \to \{0,1\}^n$.
- Given any $u \in \mathbf{F}_2^n$. can efficiently compute $f(u)$.
- Nonzero $s \in \mathbf{F}_2^n$.
- $f(u) = f(u+s)$ for all u.
- f has no other collisions.

Goal: Figure out s.

Traditional algorithm to find s: compute f for many inputs, hope to find collision.

Simon's algorithm finds s with $\approx n$ reversible computations of f.

Step 5. C_0NOT_3:

1, 0, 1, 0, 1, 0, 1, 0,
0, 1, 0, 1, 0, 1, 0, 1,
0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0.

Each column is a parallel universe performing its own computations.

Simon's algorithm

Assumptions:

- Function $f : \mathbf{F}_2^n \to \{0,1\}^n$.
- Given any $u \in \mathbf{F}_2^n$. can efficiently compute $f(u)$.
- Nonzero $s \in \mathbf{F}_2^n$.
- $f(u) = f(u+s)$ for all u.
- f has no other collisions.

Goal: Figure out s.

Traditional algorithm to find s: compute f for many inputs, hope to find collision.

Simon's algorithm finds s with $\approx n$ reversible computations of f.

Step 5b. More shuffling:

1, 0, 0, 0, 1, 0, 0, 0,
0, 1, 0, 0, 0, 1, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 1, 0, 0, 0, 1, 0,
0, 0, 0, 1, 0, 0, 0, 1,
0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0.

Each column is a parallel universe performing its own computations.

Simon's algorithm

Assumptions:

- Function $f : \mathbf{F}_2^n \to \{0,1\}^n$.
- Given any $u \in \mathbf{F}_2^n$. can efficiently compute $f(u)$.
- Nonzero $s \in \mathbf{F}_2^n$.
- $f(u) = f(u+s)$ for all u.
- f has no other collisions.

Goal: Figure out s.

Traditional algorithm to find s: compute f for many inputs, hope to find collision.

Simon's algorithm finds s with $\approx n$ reversible computations of f.

Step 5c. More shuffling:

1, 0, 0, 0, 0, 0, 0, 0,
0, 1, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 1, 0, 0, 0,
0, 0, 0, 0, 0, 1, 0, 0,
0, 0, 1, 0, 0, 0, 0, 0,
0, 0, 0, 1, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 1, 0,
0, 0, 0, 0, 0, 0, 0, 1.

Each column is a parallel universe performing its own computations.

Simon's algorithm

Assumptions:

- Function $f : \mathbf{F}_2^n \to \{0, 1\}^n$.
- Given any $u \in \mathbf{F}_2^n$, can efficiently compute $f(u)$.
- Nonzero $s \in \mathbf{F}_2^n$.
- $f(u) = f(u + s)$ for all u.
- f has no other collisions.

Goal: Figure out s.

Traditional algorithm to find s: compute f for many inputs, hope to find collision.

Simon's algorithm finds s with $\approx n$ reversible computations of f.

Step 5d. More shuffling:

1, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 1, 0, 0,
0, 0, 0, 0, 1, 0, 0, 0,
0, 1, 0, 0, 0, 0, 0, 0,
0, 0, 1, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 1,
0, 0, 0, 0, 0, 0, 1, 0,
0, 0, 0, 1, 0, 0, 0, 0.

Each column is a parallel universe performing its own computations.

Simon's algorithm

Assumptions:

- Function $f : \mathbf{F}_2^n \to \{0,1\}^n$.
- Given any $u \in \mathbf{F}_2^n$. can efficiently compute $f(u)$.
- Nonzero $s \in \mathbf{F}_2^n$.
- $f(u) = f(u+s)$ for all u.
- f has no other collisions.

Goal: Figure out s.

Traditional algorithm to find s: compute f for many inputs, hope to find collision.

Simon's algorithm finds s with $\approx n$ reversible computations of f.

Step 5e. More shuffling:
1, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 1, 0, 0,
0, 0, 0, 0, 1, 0, 0, 0,
0, 1, 0, 0, 0, 0, 0, 0,
0, 0, 1, 0, 0, 0, 0, 1,
0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 1, 0, 0, 1, 0,
0, 0, 0, 0, 0, 0, 0, 0.

Each column is a parallel universe performing its own computations.

Simon's algorithm

Assumptions:

- Function $f : \mathbf{F}_2^n \to \{0,1\}^n$.
- Given any $u \in \mathbf{F}_2^n$. can efficiently compute $f(u)$.
- Nonzero $s \in \mathbf{F}_2^n$.
- $f(u) = f(u+s)$ for all u.
- f has no other collisions.

Goal: Figure out s.

Traditional algorithm to find s: compute f for many inputs, hope to find collision.

Simon's algorithm finds s with $\approx n$ reversible computations of f.

Step 5f. More shuffling:
0, 0, 0, 0, 0, 1, 0, 0,
1, 0, 0, 0, 0, 0, 0, 0,
0, 1, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 1, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 1, 0, 0, 0, 0, 1,
0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 1, 0, 0, 1, 0.

Each column is a parallel universe performing its own computations.

Simon's algorithm

Assumptions:

- Function $f : \mathbf{F}_2^n \to \{0,1\}^n$.
- Given any $u \in \mathbf{F}_2^n$, can efficiently compute $f(u)$.
- Nonzero $s \in \mathbf{F}_2^n$.
- $f(u) = f(u+s)$ for all u.
- f has no other collisions.

Goal: Figure out s.

Traditional algorithm to find s: compute f for many inputs, hope to find collision.

Simon's algorithm finds s with $\approx n$ reversible computations of f.

Step 5g. More shuffling:

0, 1, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 1, 0, 0, 0,
0, 0, 0, 0, 0, 1, 0, 0,
1, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 1, 0, 0, 1, 0,
0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 1, 0, 0, 0, 0, 1.

Each column is a parallel universe performing its own computations.

Simon's algorithm

Assumptions:

- Function $f : \mathbf{F}_2^n \to \{0,1\}^n$.
- Given any $u \in \mathbf{F}_2^n$, can efficiently compute $f(u)$.
- Nonzero $s \in \mathbf{F}_2^n$.
- $f(u) = f(u + s)$ for all u.
- f has no other collisions.

Goal: Figure out s.

Traditional algorithm to find s: compute f for many inputs, hope to find collision.

Simon's algorithm finds s with $\approx n$ reversible computations of f.

Step 5h. More shuffling:

0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 1, 0, 0, 1, 0,
0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 1, 0, 0, 0, 0, 1,
0, 1, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 1, 0, 0, 0,
0, 0, 0, 0, 0, 1, 0, 0,
1, 0, 0, 0, 0, 0, 0, 0.

Each column is a parallel universe performing its own computations.

Simon's algorithm

Assumptions:

- Function $f : \mathbf{F}_2^n \to \{0,1\}^n$.
- Given any $u \in \mathbf{F}_2^n$.
  can efficiently compute $f(u)$.
- Nonzero $s \in \mathbf{F}_2^n$.
- $f(u) = f(u+s)$ for all u.
- f has no other collisions.

Goal: Figure out s.

Traditional algorithm to find s: compute f for many inputs, hope to find collision.

Simon's algorithm finds s with $\approx n$ reversible computations of f.

Step 5i. More shuffling:

0, 0, 0, 0, 0, 0, 1, 0,
0, 0, 0, 1, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 1,
0, 0, 1, 0, 0, 0, 0, 0,
0, 1, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 1, 0, 0, 0,
0, 0, 0, 0, 0, 1, 0, 0,
1, 0, 0, 0, 0, 0, 0, 0.

Each column is a parallel universe performing its own computations.

Simon's algorithm

Assumptions:

- Function $f : \mathbf{F}_2^n \to \{0,1\}^n$.
- Given any $u \in \mathbf{F}_2^n$. can efficiently compute $f(u)$.
- Nonzero $s \in \mathbf{F}_2^n$.
- $f(u) = f(u + s)$ for all u.
- f has no other collisions.

Goal: Figure out s.

Traditional algorithm to find s: compute f for many inputs, hope to find collision.

Simon's algorithm finds s with $\approx n$ reversible computations of f.

Step 5j. Final shuffling:

0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 1, 0, 0, 1, 0,
0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 1, 0, 0, 0, 0, 1,
0, 1, 0, 0, 1, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0,
1, 0, 0, 0, 0, 1, 0, 0.

Each column is a parallel universe performing its own computations. Now done computing $f(u)$.

Simon's algorithm

Assumptions:

- Function $f : \mathbf{F}_2^n \to \{0,1\}^n$.
- Given any $u \in \mathbf{F}_2^n$. can efficiently compute $f(u)$.
- Nonzero $s \in \mathbf{F}_2^n$.
- $f(u) = f(u+s)$ for all u.
- f has no other collisions.

Goal: Figure out s.

Traditional algorithm to find s: compute f for many inputs, hope to find collision.

Simon's algorithm finds s with $\approx n$ reversible computations of f.

Step 5j. Final shuffling:

0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 1, 0, 0, 1, 0,
0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 1, 0, 0, 0, 0, 1,
0, 1, 0, 0, 1, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0,
1, 0, 0, 0, 0, 1, 0, 0.

Each column is a parallel universe performing its own computations.
Now done computing $f(u)$.
Can see: $f(u)$ and $f(u+101)$ match.

Simon's algorithm

Assumptions:

- Function $f : \mathbf{F}_2^n \to \{0,1\}^n$.
- Given any $u \in \mathbf{F}_2^n$. can efficiently compute $f(u)$.
- Nonzero $s \in \mathbf{F}_2^n$.
- $f(u) = f(u+s)$ for all u.
- f has no other collisions.

Goal: Figure out s.

Traditional algorithm to find s: compute f for many inputs, hope to find collision.

Simon's algorithm finds s with $\approx n$ reversible computations of f.

Step 6. Hadamard$_0$:

0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 1, $\overline{1}$, 0, 0, 1, 1,
0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 1, 1, 0, 0, 1, $\overline{1}$,
1, $\overline{1}$, 0, 0, 1, 1, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0,
1, 1, 0, 0, 1, $\overline{1}$, 0, 0.

Notation: $\overline{1}$ means -1.

Simon's algorithm

Assumptions:

- Function $f : \mathbf{F}_2^n \to \{0,1\}^n$.
- Given any $u \in \mathbf{F}_2^n$. can efficiently compute $f(u)$.
- Nonzero $s \in \mathbf{F}_2^n$.
- $f(u) = f(u+s)$ for all u.
- f has no other collisions.

Goal: Figure out s.

Traditional algorithm to find s: compute f for many inputs, hope to find collision.

Simon's algorithm finds s with $\approx n$ reversible computations of f.

Step 7. Hadamard$_1$:

$0,\ 0,\ 0,\ 0,\ 0,\ 0,\ 0,\ 0,$
$1,\ \overline{1},\ \overline{1},\ 1,\ 1,\ 1,\ \overline{1},\ \overline{1},$
$0,\ 0,\ 0,\ 0,\ 0,\ 0,\ 0,\ 0,$
$1,\ 1,\ \overline{1},\ \overline{1},\ 1,\ \overline{1},\ \overline{1},\ 1,$
$1,\ \overline{1},\ 1,\ \overline{1},\ 1,\ 1,\ 1,\ 1,$
$0,\ 0,\ 0,\ 0,\ 0,\ 0,\ 0,\ 0,$
$0,\ 0,\ 0,\ 0,\ 0,\ 0,\ 0,\ 0,$
$1,\ 1,\ 1,\ 1,\ 1,\ \overline{1},\ 1,\ \overline{1}.$

Simon's algorithm

Assumptions:

- Function $f : \mathbf{F}_2^n \to \{0,1\}^n$.
- Given any $u \in \mathbf{F}_2^n$. can efficiently compute $f(u)$.
- Nonzero $s \in \mathbf{F}_2^n$.
- $f(u) = f(u+s)$ for all u.
- f has no other collisions.

Goal: Figure out s.

Traditional algorithm to find s: compute f for many inputs, hope to find collision.

Simon's algorithm finds s with $\approx n$ reversible computations of f.

Step 8. Hadamard$_2$:

$0, 0, 0, 0, 0, 0, 0, 0,$
$2, 0, \overline{2}, 0, 0, \overline{2}, 0, 2,$
$0, 0, 0, 0, 0, 0, 0, 0,$
$2, 0, \overline{2}, 0, 0, 2, 0, \overline{2},$
$2, 0, 2, 0, 0, \overline{2}, 0, \overline{2},$
$0, 0, 0, 0, 0, 0, 0, 0,$
$0, 0, 0, 0, 0, 0, 0, 0,$
$2, 0, 2, 0, 0, 2, 0, 2.$

Simon's algorithm

Assumptions:

- Function $f : \mathbf{F}_2^n \to \{0,1\}^n$.
- Given any $u \in \mathbf{F}_2^n$. can efficiently compute $f(u)$.
- Nonzero $s \in \mathbf{F}_2^n$.
- $f(u) = f(u+s)$ for all u.
- f has no other collisions.

Goal: Figure out s.

Traditional algorithm to find s: compute f for many inputs, hope to find collision.

Simon's algorithm finds s with $\approx n$ reversible computations of f.

Step 8. Hadamard$_2$:

0, 0, 0, 0, 0, 0, 0, 0,
2, 0, $\overline{2}$, 0, 0, $\overline{2}$, 0, 2,
0, 0, 0, 0, 0, 0, 0, 0,
2, 0, $\overline{2}$, 0, 0, 2, 0, $\overline{2}$,
2, 0, 2, 0, 0, $\overline{2}$, 0, $\overline{2}$,
0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0,
2, 0, 2, 0, 0, 2, 0, 2.

Step 9: Measure first 3 qubits. Obtain some information about "period" s: a random vector orthogonal to 101.

Simon's algorithm

Assumptions:

- Function $f : \mathbf{F}_2^n \rightarrow \{0,1\}^n$.
- Given any $u \in \mathbf{F}_2^n$.
  can efficiently compute $f(u)$.
- Nonzero $s \in \mathbf{F}_2^n$.
- $f(u) = f(u+s)$ for all u.
- f has no other collisions.

Goal: Figure out s.

Traditional algorithm to find s: compute f for many inputs, hope to find collision.

Simon's algorithm finds s with $\approx n$ reversible computations of f.

Step 8. Hadamard$_2$:

$0,\ 0,\ 0,\ 0,\ 0,\ 0,\ 0,\ 0,$
$2,\ 0,\ \overline{2},\ 0,\ 0,\ \overline{2},\ 0,\ 2,$
$0,\ 0,\ 0,\ 0,\ 0,\ 0,\ 0,\ 0,$
$2,\ 0,\ \overline{2},\ 0,\ 0,\ 2,\ 0,\ \overline{2},$
$2,\ 0,\ 2,\ 0,\ 0,\ \overline{2},\ 0,\ \overline{2},$
$0,\ 0,\ 0,\ 0,\ 0,\ 0,\ 0,\ 0,$
$0,\ 0,\ 0,\ 0,\ 0,\ 0,\ 0,\ 0,$
$2,\ 0,\ 2,\ 0,\ 0,\ 2,\ 0,\ 2.$

Step 9: Measure first 3 qubits.
Obtain some information about "period" s:
a random vector orthogonal to 101.
Repeat to pin down 101

Generalizations

Generalize Step 5 to any function : $\mathbf{F}_2^n \to \{0,1\}^m$
which satisfies $f(u) = f(u+s)$ for some s.
"Usually" algorithm figures out s.

Generalizations

Generalize Step 5 to any function : $\mathbf{F}_2^n \to \{0,1\}^m$
which satisfies $f(u) = f(u+s)$ for some s.
"Usually" algorithm figures out s.

Shor's algorithm replaces addition in $\mathbf{F}_2^n$ with more general $+$ operation.
Many spectacular applications.

Generalizations

Generalize Step 5 to any function : $\mathbf{F}_2^n \to \{0,1\}^m$
which satisfies $f(u) = f(u+s)$ for some s.
"Usually" algorithm figures out s.

Shor's algorithm replaces addition in $\mathbf{F}_2^n$ with more general + operation.
Many spectacular applications.

e.g. Shor finds "random" s with

$$2^u \equiv 2^{u+s} \bmod N.$$

Easy to factor N using this. (See video Shor vs. RSA.)

Generalizations

Generalize Step 5 to any function : $\mathbf{F}_2^n \to \{0,1\}^m$
which satisfies $f(u) = f(u+s)$ for some s.
"Usually" algorithm figures out s.

Shor's algorithm replaces addition in $\mathbf{F}_2^n$ with more general $+$ operation.
Many spectacular applications.

e.g. Shor finds "random" s with

$$2^u \equiv 2^{u+s} \bmod N.$$

Easy to factor N using this. (See video Shor vs. RSA.)

e.g. Shor finds "random" s, t with

$$g^u h^v \equiv g^{u+s} h^{v+t} \bmod p.$$

Easy to compute discrete logs: $\log_g(h) \equiv -s/t \bmod \ell$, where $\ell = \text{ord}(g)$.