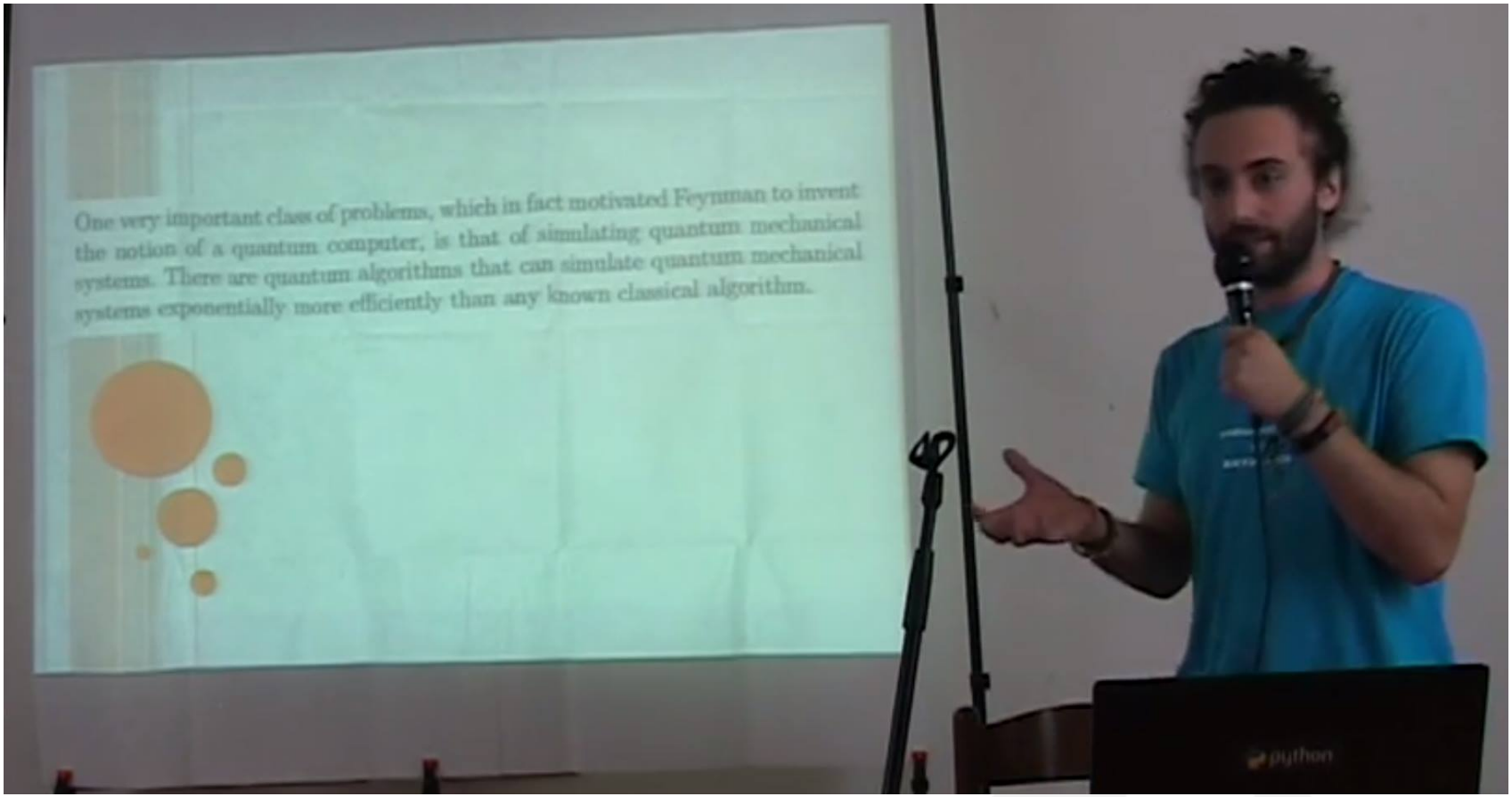
Quantum tools for cybersecurity

Algorithms and protocols for defense

Italian Hacker Camp - Padua

August 3, 2018

Twitter: @scinawa
https://luongo.pro



What have we said so far?

Cool work is always result of a great team

- Iordanis Kerenidis (work on QSFA)
- Bull/Atos' team (computational power & useful discussions)
- Nicola Gigante (discussion on Büchi automata)
- Frédéric Magniez, Anupam Prakash, Jonas Landman (useful discussions)
- Luis Trigo Vidarte & Mathieu Bozzio (the pictures @ LIP6).

Introduction to quantum computation

Amplitude amplification and language security

Quantum machine learning on security datasets

Quantum algorithms for formal software verification

Further impacts con information security

- Information-theoretically secure Key-Distribution
- Blind quantum computation
- Position based quantum cryptography

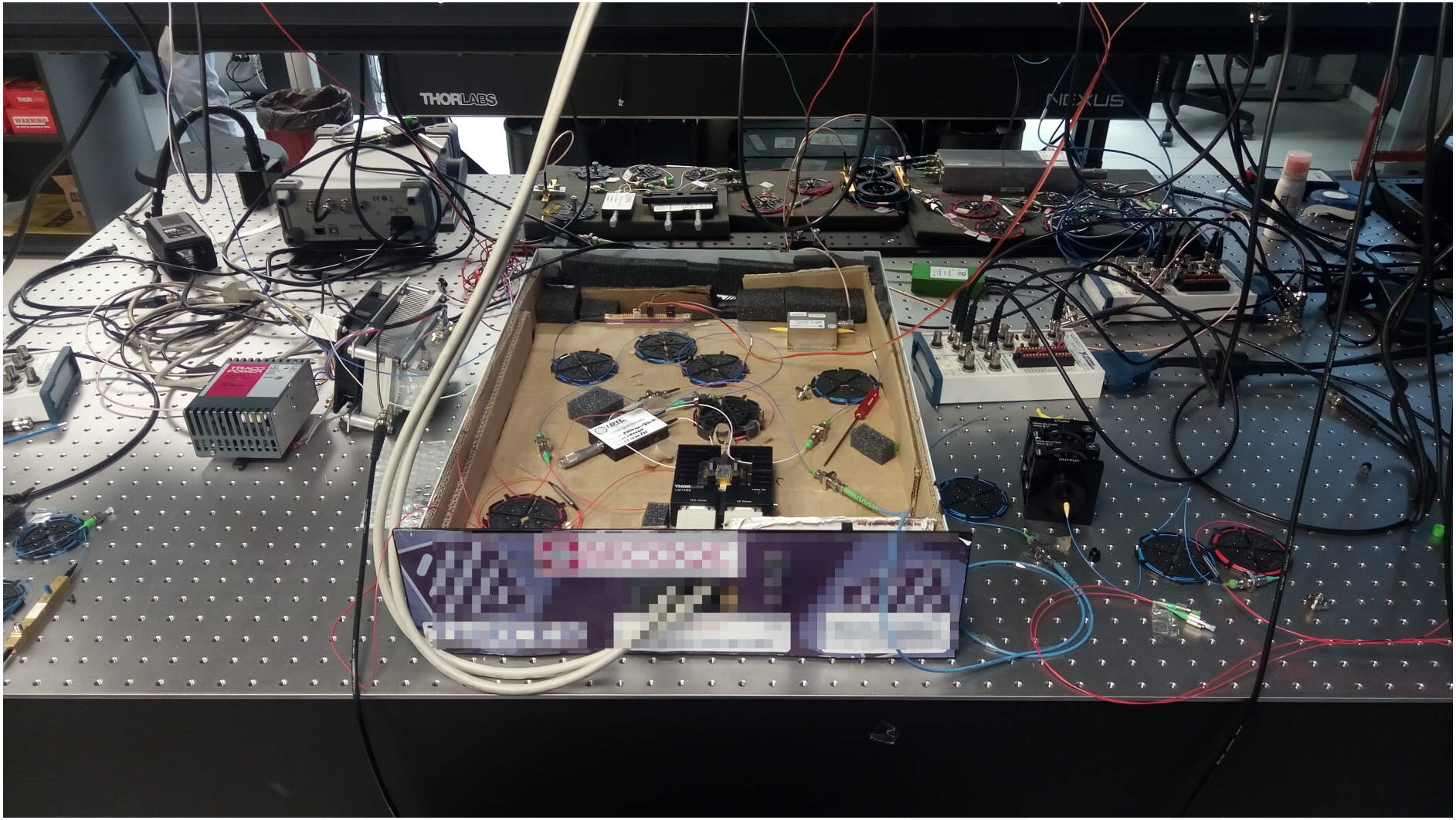


Figure: What is this? (Courtesy of LIP6)[BOV+18]

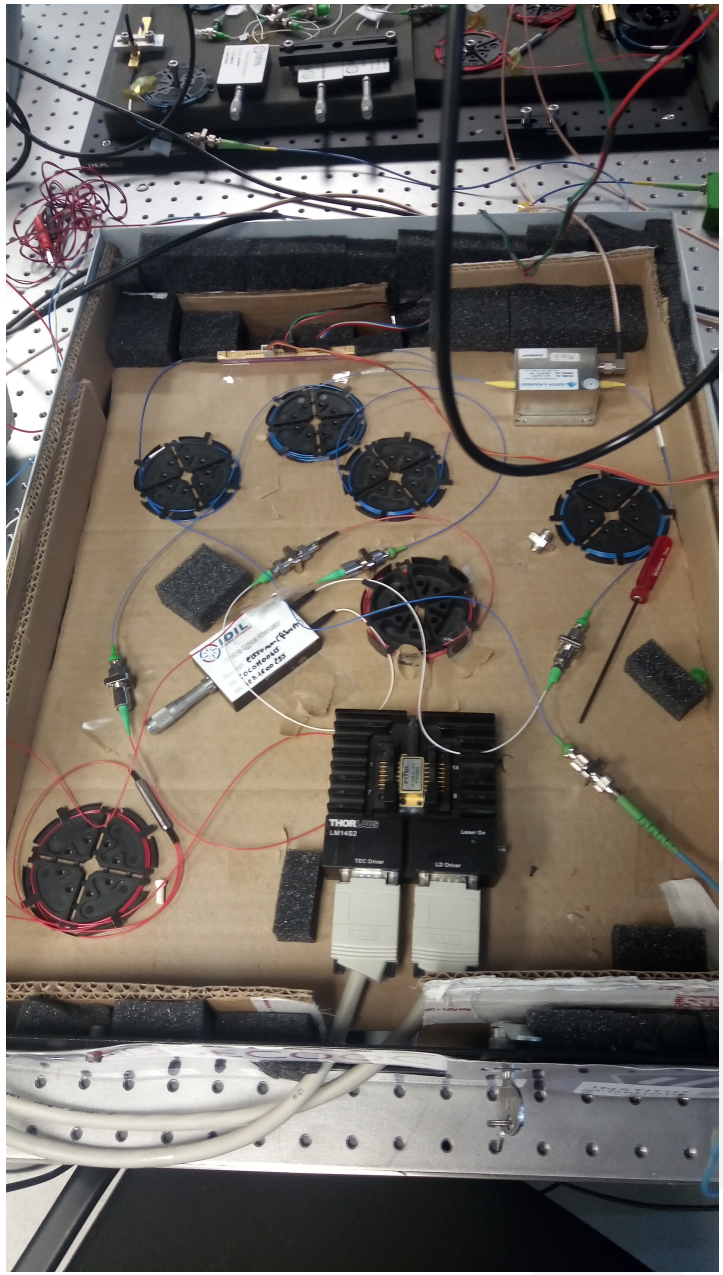
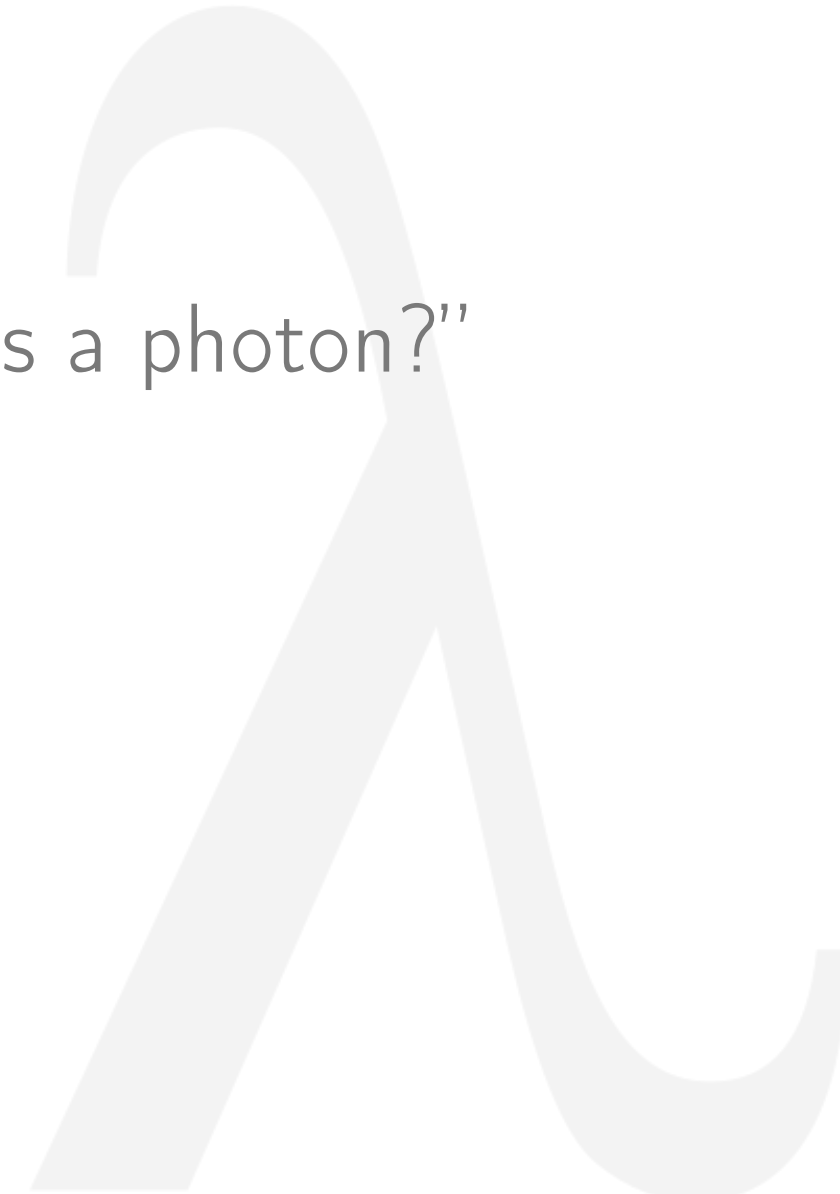


Figure: What is this? (Courtesy of LIP6) [BOV+18]

Figure: What is this?(Courtesy of LIP6)

Figure: What is this?(Courtesy of LIP6)

“Scinawa, WTF is a photon?”



I know what a qubit is!

- A qubit is a unitary vector in $\mathbb{C}^2$.
- Example: $|\psi\rangle = [a, b]$ s.t. $|a|^2 + |b|^2 = 1$ and $a, b \in \mathbb{C}$
- These 2 vectors form a base, and encode 0 and 1:

$0 \Rightarrow |0\rangle = [1, 0], \quad 1 \Rightarrow |1\rangle = [0, 1]$

- $|\psi_{\mathsf{CAT}}\rangle = \frac{1}{\sqrt{2}}|0\rangle + \frac{1}{\sqrt{2}}|1\rangle$

$|\psi_{\mathsf{CAT}}\rangle = \frac{1}{\sqrt{2}}|\mathsf{DEAD}\rangle + \frac{1}{\sqrt{2}}|\mathsf{ALIVE}\rangle$

I know what a qubit is!

- A qubit is a unitary vector in $\mathbb{C}^2$.
- Example: $|\psi\rangle = [a, b]$ s.t. $|a|^2 + |b|^2 = 1$ and $a, b \in \mathbb{C}$
- These 2 vectors form a base, and encode 0 and 1:

$0 \Rightarrow |0\rangle = [1, 0], \quad 1 \Rightarrow |1\rangle = [0, 1]$

- $|\psi_{\mathsf{CAT}}\rangle = \frac{1}{\sqrt{2}}|0\rangle + \frac{1}{\sqrt{2}}|1\rangle$

$|\psi_{\mathsf{CAT}}\rangle = \frac{1}{\sqrt{2}}|\mathsf{DEAD}\rangle + \frac{1}{\sqrt{2}}|\mathsf{ALIVE}\rangle$

I know what a qubit is!

- A qubit is a unitary vector in $\mathbb{C}^2$.
- Example: $|\psi\rangle = [a, b]$ s.t. $|a|^2 + |b|^2 = 1$ and $a, b \in \mathbb{C}$
- These 2 vectors form a base, and encode 0 and 1:

$0 \Rightarrow |0\rangle = [1, 0], \quad 1 \Rightarrow |1\rangle = [0, 1]$

- $|\psi_{\mathsf{CAT}}\rangle = \frac{1}{\sqrt{2}}|0\rangle + \frac{1}{\sqrt{2}}|1\rangle$

$|\psi_{\mathsf{CAT}}\rangle = \frac{1}{\sqrt{2}}|\mathsf{DEAD}\rangle + \frac{1}{\sqrt{2}}|\mathsf{ALIVE}\rangle$

I know what a qubit is!

- A qubit is a unitary vector in $\mathbb{C}^2$.
- Example: $|\psi\rangle = [a, b]$ s.t. $|a|^2 + |b|^2 = 1$ and $a, b \in \mathbb{C}$
- These 2 vectors form a base, and encode 0 and 1:

$0 \Rightarrow |0\rangle = [1, 0], \quad 1 \Rightarrow |1\rangle = [0, 1]$

- $|\psi_{\mathsf{CAT}}\rangle = \frac{1}{\sqrt{2}}|0\rangle + \frac{1}{\sqrt{2}}|1\rangle$

$|\psi_{\mathsf{CAT}}\rangle = \frac{1}{\sqrt{2}}|\mathsf{DEAD}\rangle + \frac{1}{\sqrt{2}}|\mathsf{ALIVE}\rangle$

I know what a qubit is!

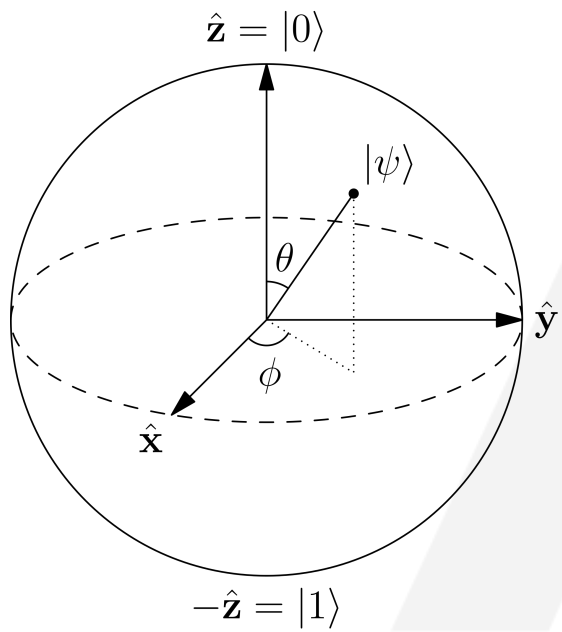
- A qubit is a unitary vector in $\mathbb{C}^2$.
- Example: $|\psi\rangle = [a, b]$ s.t. $|a|^2 + |b|^2 = 1$ and $a, b \in \mathbb{C}$
- These 2 vectors form a base, and encode 0 and 1:

$0 \Rightarrow |0\rangle = [1, 0]^T, \quad 1 \Rightarrow |1\rangle = [0, 1]^T$

- $|\psi_{\mathsf{CAT}}\rangle = \frac{1}{\sqrt{2}}|0\rangle + \frac{1}{\sqrt{2}}|1\rangle$

$|\psi_{\mathsf{CAT}}\rangle = \frac{1}{\sqrt{2}}|\mathsf{DEAD}\rangle + \frac{1}{\sqrt{2}}|\mathsf{ALIVE}\rangle$

Figure: Bloch's sphere representation of a qubit



Other axioms...

Composition of multiple qubits using tensor product:

$$|\psi_1\rangle \otimes |\psi_2\rangle = [a, b] \otimes [c, d] = [ac, ad, bc, bd]$$

.

Takeaway: With n qubits we describe state

$$|\psi\rangle = \sum_{i=0}^{2^n-1} \alpha_i |i\rangle$$

with $\sum_{i=0}^{2^n-1} |\alpha_i|^2 = 1$.

- Quantum state $\propto$ probability distribution...

- Unitary vector means that $\langle\psi|\psi\rangle = 1$

Other axioms...

Composition of multiple qubits using tensor product:

$$|\psi_1\rangle \otimes |\psi_2\rangle = [a, b] \otimes [c, d] = [ac, ad, bc, bd]$$

.

Takeaway: With n qubits we describe state

$$|\psi\rangle = \sum_{i=0}^{2^n-1} \alpha_i |i\rangle$$

with $\sum_{i=0}^{2^n-1} |\alpha_i|^2 = 1$.

- Quantum state $\propto$ probability distribution...

- Unitary vector means that $\langle\psi|\psi\rangle = 1$

Other axioms...

Composition of multiple qubits using tensor product:

$$|\psi_1\rangle \otimes |\psi_2\rangle = [a, b] \otimes [c, d] = [ac, ad, bc, bd]$$

.

Takeaway: With n qubits we describe state

$$|\psi\rangle = \sum_{i=0}^{2^n-1} \alpha_i |i\rangle$$

with $\sum_{i=0}^{2^n-1} |\alpha_i|^2 = 1$.

- Quantum state $\propto$ probability distribution..

- Unitary vector means that $\langle\psi|\psi\rangle = 1$

Other axioms...

Composition of multiple qubits using tensor product:

$$|\psi_1\rangle \otimes |\psi_2\rangle = [a, b] \otimes [c, d] = [ac, ad, bc, bd]$$

.

Takeaway: With n qubits we describe state

$$|\psi\rangle = \sum_{i=0}^{2^n-1} \alpha_i |i\rangle$$

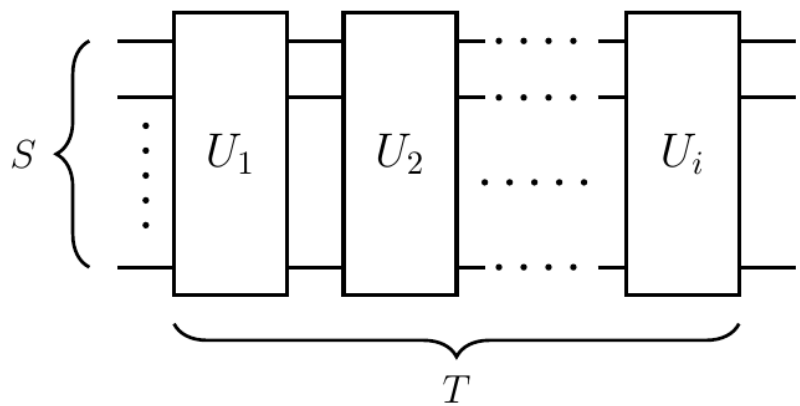
with $\sum_{i=0}^{2^n-1} |\alpha_i|^2 = 1$.

- Quantum state $\propto$ probability distribution..
- Unitary vector means that $\langle\psi|\psi\rangle = 1$

Other axioms...

- Computation is performed by gates(= **matrices**) that act on qubits.
- (many) set of gates universal for **classical** computation
- (many) sets of gates universal for **quantum** computation :)
- A gate/matrix U is unitary iff $UU^* = U^*U = I$. (unitary = quantum)
- We can always apply the **inverse** of a gate

Figure: A quantum circuit



Quantum gates: example

$$X = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}$$

$$X|0\rangle = |1\rangle \quad X|1\rangle = |0\rangle$$

$$CNOT = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{bmatrix}$$

$$CNOT\,|0\rangle\,|0\rangle = |0\rangle\,|0\rangle$$

$$CNOT\,|1\rangle\,|0\rangle = |1\rangle\,|1\rangle$$

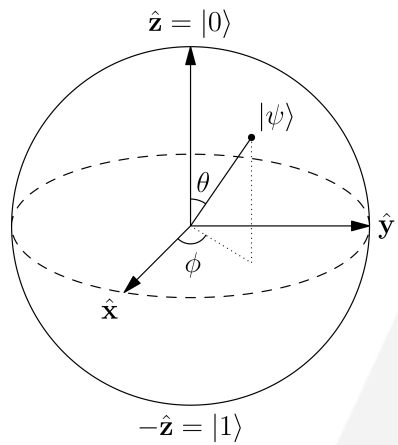
Quantum gates: more quantumish examples

$$H = \frac{1}{\sqrt{2}} \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}$$

$$H|0\rangle = |+\rangle = \frac{1}{\sqrt{2}}|0\rangle + \frac{1}{\sqrt{2}}|1\rangle$$

$$H|1\rangle = |-\rangle = \frac{1}{\sqrt{2}}|0\rangle - \frac{1}{\sqrt{2}}|1\rangle$$

Figure: Bloch's sphere representation of a qubit



Theorem

There is no unitary U_{copy} such that $U|\psi\rangle|0\rangle \rightarrow |\psi\rangle|\psi\rangle$

Proof.

Suppose U_{copy} exist. Then we could write:

$$U_{copy}|\psi_1\rangle|\alpha\rangle = |\psi_1\rangle|\psi_1\rangle \quad U_{copy}|\psi_2\rangle|\alpha\rangle = |\psi_2\rangle|\psi_2\rangle$$

We check if U_{copy} preserve the norm:

$$\langle\alpha|\langle\psi_1|U_{copy}^{\dagger}U_{copy}|\psi_2\rangle|\alpha\rangle = \langle\psi_1|\langle\psi_1||\psi_2\rangle|\psi_2\rangle$$

Since $U_{copy}^{\dagger}U = I$ we can write

$$\langle\alpha|\alpha\rangle\langle\psi_1|\psi_2\rangle = |\langle\psi_1|\psi_2\rangle|^2 \Rightarrow \langle\psi_1|\psi_2\rangle = |\langle\psi_1|\psi_2\rangle|^2$$

Therefore, if U_{copy} exist is not unitary and therefore not a valid quantum mechanic operation.. maybe magic!

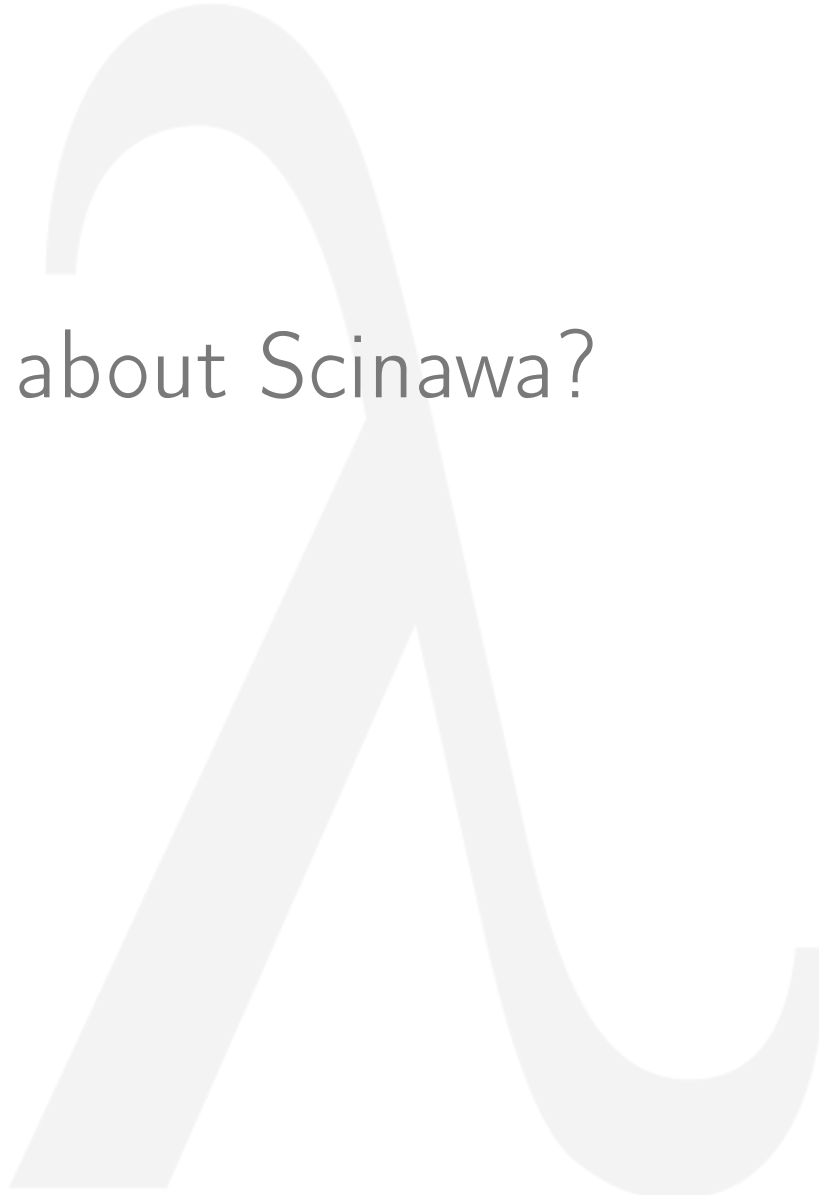
□

STDOUT for a quantum computer

- A POVM is a collection of positive operators M_i such that $\sum_i M_i = I_{2^n}$.
- The probability of reading a certain outcome given a quantum state $|\phi\rangle$ is $\langle\phi| M_i |\phi\rangle$.
- Let $|\psi\rangle = \sum_{i=0}^{2^n-1} \alpha_i |i\rangle$.

If the state is measured in the standard basis, ($M_i = |e_i\rangle \langle e_i|$), the probability of measuring i-th outcome is given by $|\alpha_i|^2$.

WTF are you talking about Scinawa?



PoC || GTFO

Quantum computers as GPUs

```
a = 4
b = 4

n = max([len(bin(a)[2:]), len(bin(b)[2:]) ])+3

print(n)
program_types_spec = {
    "circuits": [{
        "name": "Quantum-Adder-circuit",
        "quantum_registers": [

        { "name": "ripple",
          "size": 1},

       { "name": "a",
          "size": n},

        { "name": "b",
          "size": n},

        { "name": "carry",
          "size": 1}

        ],
        "classical_registers": [
            {"name": "sum",
              "size": n},
            {"name": "carrysum",
            "size": 1},

        ]
    }],
}
```

Figure: Define our quantum program

```
qp = QuantumProgram(program_types_spec)
qc = qp.get_circuit("Quantum-Adder-circuit")

qr_b = qp.get_quantum_register("b")
qr_a = qp.get_quantum_register("a")
q_ripple = qp.get_quantum_register("ripple")
q_carry = qp.get_quantum_register("carry")
c_sum = qp.get_classical_register("sum")
c_carrysum = qp.get_classical_register("carrysum")


qmalt.utils.initialize_index(qc, qr_a, a)
qmalt.utils.initialize_index(qc, qr_b, b)

Adder.apply(qc, q_ripple, qr_a, qr_b,  q_carry)

qc.measure(qr_b, c_sum)
```

Figure: Set values of quantum registers and do the sum :)

```
def apply(self, ripple, a, b, carry):
    """
    A simple ripple carry adder (no optimizations)

    :param self: quantum circuit
    :param ripple: quantum register of 1 qubit
    :param x: n qubit quanntum register
    :param y: n qubit quantum register
    :param carry: quantum register of 1 qubit
    :return: None
    """
    assert a.size == b.size
    assert ripple.size == carry.size == 1

    n = a.size
    majority(self, ripple[0], b[0], a[0])

    for j in range(n - 1):
        print("1){} a[{}], b[{}], a[{}]".format(j, j, j+1, j+1))
        majority(self, a[j], b[j+1], a[j+1])

    self.cx(a[a.size-1], carry[0])

    for j in range(n - 2, -1, -1):
        print("2){} a[{}], b[{}], a[{}]".format(j, j, j+1, j+1))
        unmajority(self, a[j], b[j+1], a[j+1])

    unmajority(self, ripple[0], b[0], a[0])
```

Figure: The "real" code for summing registers

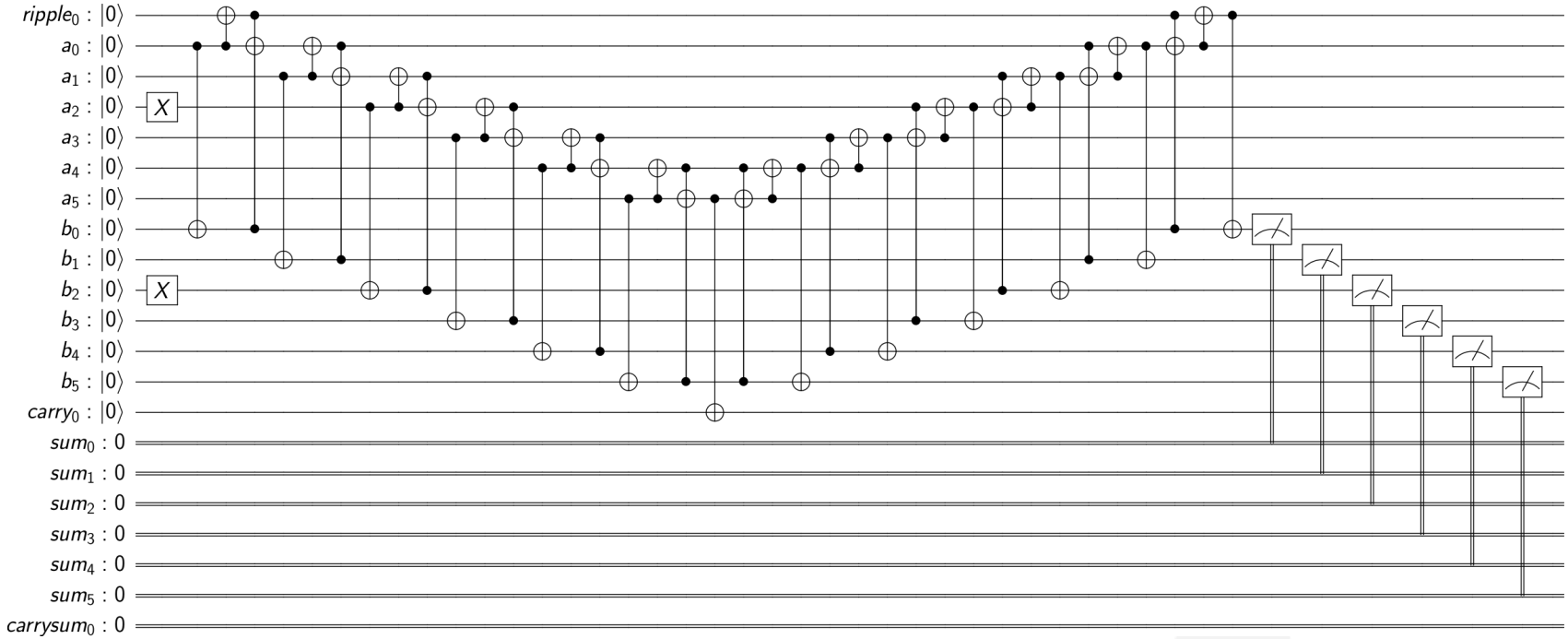


Figure: How quantum code looks like.

Quantum Assembly:

X a[2]

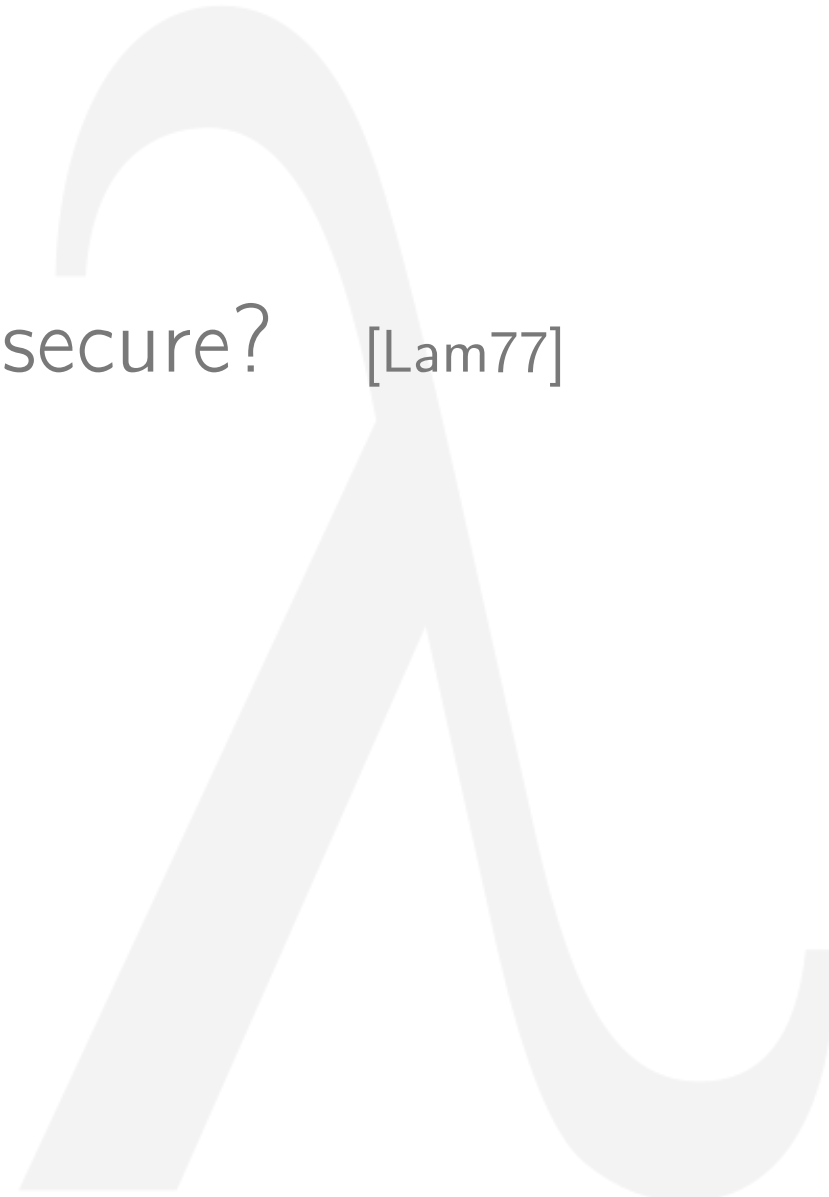
X b[2]

CNOT a[0], b[0]

...

(Quantum) Language Security

When a program is secure? [Lam77]

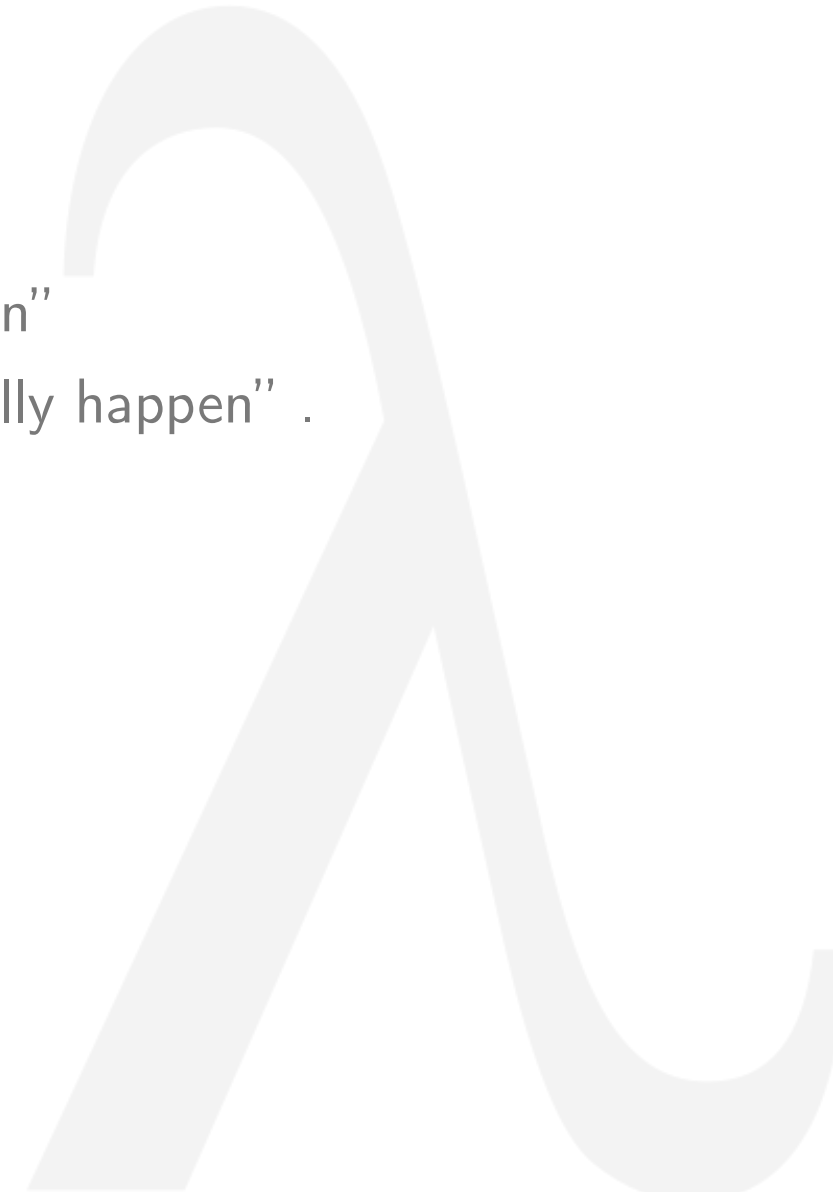


What is satefy?

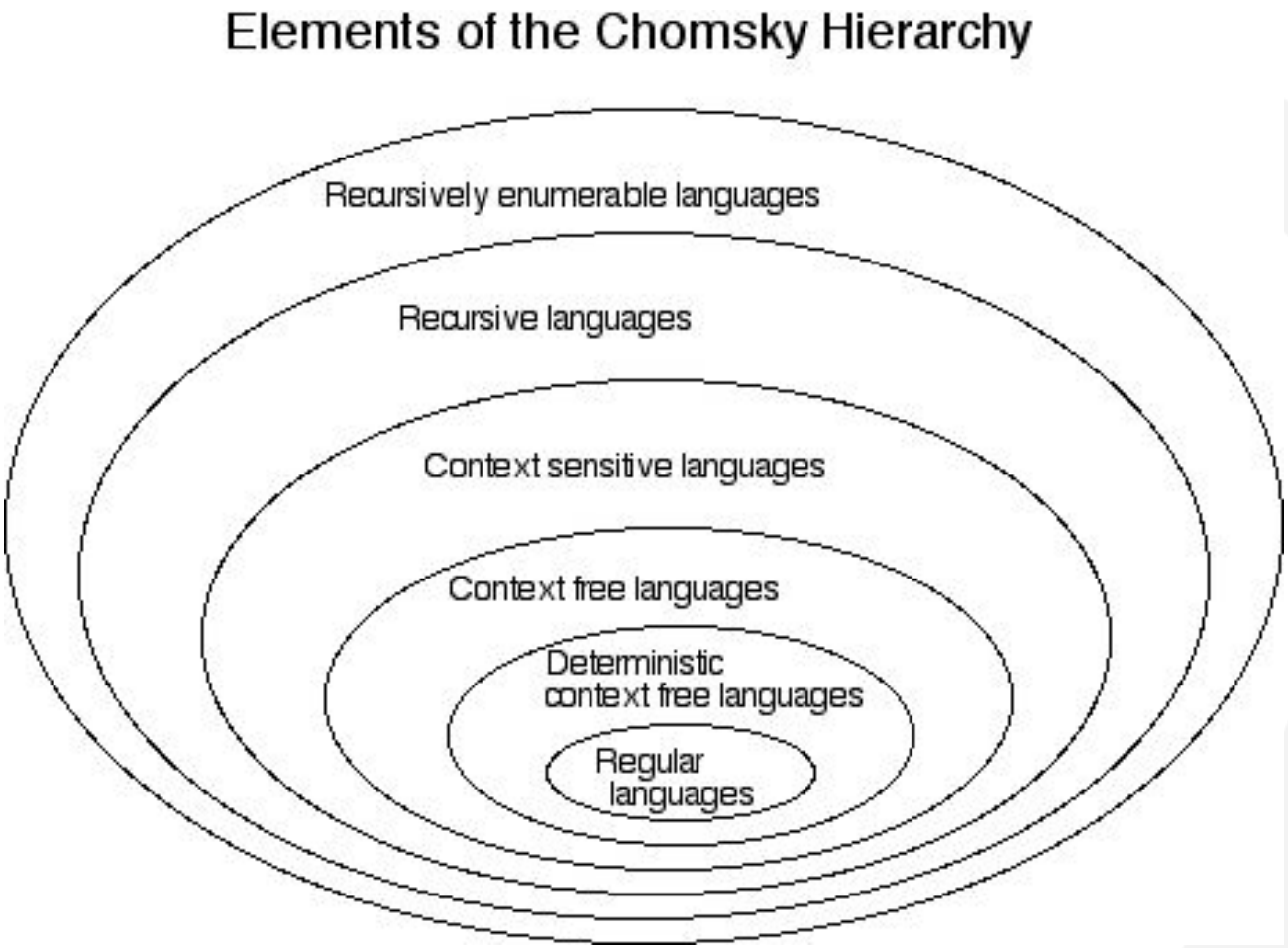
What is liveness?

Satefy = “bad things will not happen”

Liveness = “good thing will eventually happen” .



Formal languages



See [SPBL13]
Sassaman, Len, et al. "Security applications of formal language theory."
IEEE Systems Journal 7.3 (2013): 489-500.

Parser properties

(i) *soundness* (safety) - a parser accept only string in the corresponding language and rejects everything else,

(ii) *termination* (safety) - a parser eventually halts on every string presented to it,

(iii) *completeness* (liveness) - a complete parser accepts every string in its corresponding language.

These properties derive from the fact languages up to deterministic context-free can be parsed by a deterministic automata.

Threat Model

Suppose you are a service provider that has to process a certain number of message per second: $s_1, ...s_N$. The messages for the server are expected to be words of a formal language $\mathcal{L}$ that can be specified by a grammar G for which a deterministic parser exist (i.e. deterministic context free grammar). The server is expected to behave correctly on inputs in $\mathcal{L}$. An input is considered malicious if it is not in $\mathcal{L}$. We assume to have quantum access to those messages. We are tasked to find the malicious inputs in the dataset, such that we can remove them, and send the good ones to the service provider.

How much does it takes classically to test all the inputs?

Threat Model

Suppose you are a service provider that has to process a certain number of message per second: $s_1, ...s_N$. The messages for the server are expected to be words of a formal language $\mathcal{L}$ that can be specified by a grammar G for which a deterministic parser exist (i.e. deterministic context free grammar). The server is expected to behave correctly on inputs in $\mathcal{L}$. An input is considered malicious if it is not in $\mathcal{L}$. We assume to have quantum access to those messages. We are tasked to find the malicious inputs in the dataset, such that we can remove them, and send the good ones to the service provider.

How much does it takes classically to test all the inputs?

Theorem (Amplitude amplification)

[BHMT02] Let A be any quantum algorithm that uses no measurements, and let $f : \{0,1\}^n \to \{0,1\}$ be any Boolean function. There exists a quantum algorithm that given the initial success probability $a > 0$ of A, finds a good solution with certainty using a number of applications of A and A^{-1} which is in $\Theta(1/\sqrt{a})$ in the worst case.

$$A\frac{1}{\sqrt{N}}\sum_{i=0}^{N}|i\rangle|0\rangle \rightarrow \frac{1}{\sqrt{N}}\sum_{i=0}^{N}|i\rangle|s_i\rangle = \frac{1}{\sqrt{N}}\left(\sum_{i\ s.t.\ s_i\in L}|i\rangle|s_i\rangle + \sum_{i\ s.t.\ s_i\notin \mathcal{L}}|i\rangle|s_i\rangle\right)$$

$$|\psi_G\rangle := \frac{1}{\sqrt{N-m}}\sum_{i\in\mathbb{L}}|s_i\rangle \quad \text{and} \quad |\psi_M\rangle := \frac{1}{\sqrt{m}}\sum_{i\notin\mathbb{L}}|s_i\rangle$$

We can rewrite it's output state as follow:

$$|\psi\rangle = \sqrt{\frac{m}{N}}|\psi_M\rangle + \sqrt{\frac{N-m}{N}}|\psi_G\rangle = |\psi\rangle = sin(\theta)|\psi_M\rangle + cos(\theta)|\psi_G\rangle$$

Building U_f from f

$$U_f |s_i\rangle |0\rangle = \begin{cases} |s_i\rangle |1\rangle, & \text{if } s_i \notin L \\ |s_i\rangle |0\rangle, & \text{if } s_i \in L \end{cases}$$

Is just the quantum circuit implementing the classical function!

Building U_f from f

$$U_f |s_i\rangle |0\rangle = \begin{cases} |s_i\rangle |1\rangle, & \text{if } s_i \notin L \\ |s_i\rangle |0\rangle, & \text{if } s_i \in L \end{cases}$$

Is just the quantum circuit implementing the classical function!

Amplitude amplification...in practice!

We defined

$$|\psi\rangle = sin(\theta)\,|\psi_M\rangle + cos(\theta)\,|\psi_G\rangle$$

Let $Q = AU_{0\perp}A^{-1}U_p$.

$$Q^k\,|\psi\rangle = cos((2k+1)\theta)\,|\psi_m\rangle + sin((2k+1)\theta)\,|\psi_g\rangle \tag{1}$$

Quantum LangSec Firewall (with sampling)

Require:

Access to oracle A storing N input string

A grammar $\mathcal{G}$ of a LangSec language L, error parameter ε.

Ensure:

Our data contains only safe input

1: Construct a quantum circuit U_f for parsing G
2: **for** $k \in \{0 \cdots 4 \log \epsilon\}$ **do**:
3: **for** $j \in \{0 \cdots \lceil \log_c(1/sin(2sin^{-1}(1/\sqrt{N}))) \rceil\}$ **do**:
4: Extract $m \in [\, \lceil c^j \rceil \,]$
5: Create $|\varphi\rangle = A \sum_i |i\rangle |0\rangle$ and perform $Q^m |\varphi\rangle$
6: Measure the index register $\rightarrow i$.
7: **if** $s_i \notin \mathcal{L}$ **then**
8: Remove the sample from the database: $A = A/\{s_i\}$
9: **end if**
10: **end for**
11: **end for**

Theorem (Quantum firewall)

Given quantum access to N different words of length up to l and the specification of a deterministic context free grammar G there exist an algorithm that removes strings that don't belong to the language $\mathcal{L}(G)$ with certainty, and the expected number of evaluations of U_f and A in $\Theta\left(\sqrt{mN}\right)$.

(classically is $O(N)$)

Surveillance anyone?

Code

XML? Snort regexp?

Resource estimation

Given a grammar $\mathcal{G}$, for which N do we get an advantage? [ABL$^+$17]

Quantum Machine Learning (for fun and profit)

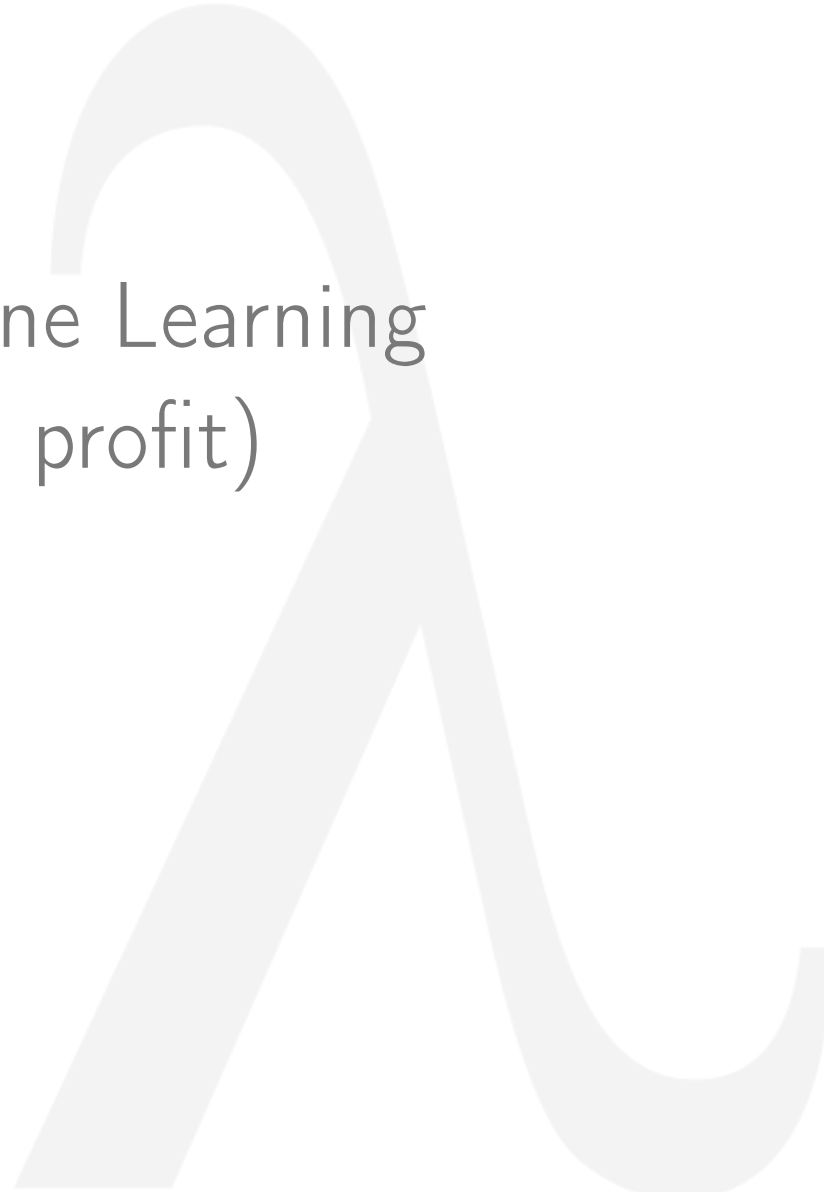


Figure: Scinawa in Erasmus @ Bochum, with <3

Slow feature analysis

Training set $X \in \mathbb{R}^{n \times d}$

$$x(i) = [x_1(i)), \cdots, x_d(i)] \in \mathbb{R}^d, i \in [n]$$

.
Each $x(i)$ belongs to one of K different classes.
The goal is to learn $K-1$ functions $g_j(x(i)), j \in [K-1]$ such that:

$$y(i) = [g_1(x(i)), \cdots, g_{K-1}(x(i))]$$

is very similar for the training samples of the same class and largely different for samples of different classes.

Slow feature analysis

Let $a = \sum_{k=1}^{K} \binom{|T_k|}{2}$. $\forall\, j \in [K-1]$, minimize:

$$\Delta(y_j) = \frac{1}{a} \sum_{k=1}^{K} \sum_{\substack{s,t \in T_k \\ s<t}} (g_j(x(s)) - g_j(x(t)))^2$$

with the following constraints:

$$\frac{1}{n} \sum_{k=1}^{K} \sum_{i \in T_k} g_j(x(i)) = 0 \quad \forall j \in [K-1]$$

$$\frac{1}{n} \sum_{k=1}^{K} \sum_{i \in T_k} g_j(x(i))^2 = 1 \quad \forall j \in [K-1]$$

$$\frac{1}{n} \sum_{k=1}^{K} \sum_{i \in T_k} g_j(x(i)) g_v(x(i)) = 0 \quad \forall v < j \in [K-1]$$

Slow feature analysis

Let $a = \sum_{k=1}^{K} \binom{|T_k|}{2}$. $\forall\, j \in [K-1]$, minimize:

$$\Delta(y_j) = \frac{1}{a} \sum_{k=1}^{K} \sum_{\substack{s,t \in T_k \\ s<t}} (g_j(x(s)) - g_j(x(t)))^2$$

with the following constraints:

- zero mean
- unitary variance
- signals are decorrelated

Matrix algebra

Theorem ([CGJ18, GSLW18])

Let $M := \sum_i \sigma_i u_i v_i^T \in \mathbb{R}^{d\times d}$ such that $\|M\|_2 = 1$, and a vector $x \in \mathbb{R}^d$ stored in QRAM. There exist quantum algorithms that with probability at least $1 - 1/poly(d)$ return:

- *a state $|z\rangle$ such that $\| |z\rangle - |Mx\rangle \| \leq \epsilon$ in time $\tilde{O}(\kappa(M)\mu(M)\log(1/\epsilon))$*
- *a state $|z\rangle$ such that $\| |z\rangle - |M^{-1}x\rangle \| \leq \epsilon$ in time $\tilde{O}(\kappa(M)\mu(M)\log(1/\epsilon))$*
- *a state $|M^+_{\leq\theta,\delta}M_{\leq\theta,\delta}x\rangle$ in time $\tilde{O}(\frac{\mu(M)\|x\|}{\delta\theta\left\|M^+_{\leq\theta,\delta}M_{\leq\theta,\delta}x\right\|})$*

One can also get estimates of the norms with multiplicative error η by increasing the running time by a factor $1/\eta$.

Theorem (QSFA algorithm [KL18])

Let $X = \sum_i \sigma_i u_i v_i^T \in \mathbb{R}^{n \times d}$ and its derivative matrix $\dot{X} \in \mathbb{R}^{n \log n \times d}$ stored in QRAM format. Let $\epsilon, \theta, \delta, \eta > 0$. There exists a quantum algorithm that produces as output a state $|\overline{Y}\rangle$ with $| |\overline{Y}\rangle - |A^+_{\leq \theta,\delta} A_{\leq \theta,\delta} Z\rangle | \leq \epsilon$ in time $\tilde{O}\left(\left(\kappa(X)\mu(X)\log(1/\varepsilon) + \frac{(\kappa(X)+\kappa(\dot{X}))(\mu(X)+\mu(\dot{X}))}{\delta\theta}\right)\frac{||Z||}{||A^+_{\leq \theta,\delta} A_{\leq \theta,\delta} Z||}\right)$ and an estimator $\overline{\|Y\|}$ with $|\overline{\|Y\|} - \|Y\|| \leq \eta \|Y\|$ with an additional $1/\eta$ factor.

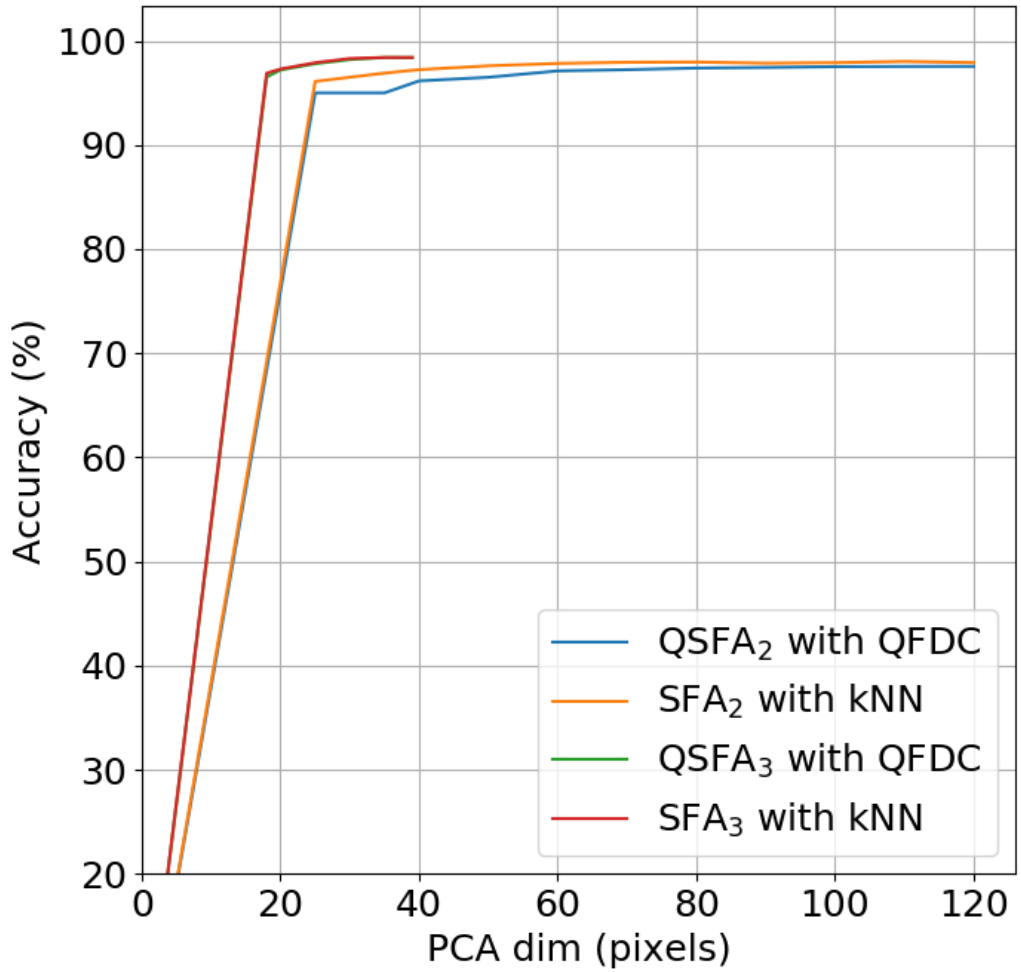


Figure: Comparing QSFA and SFA on MNIST

SPAM: notes in quantum machine learning

return λ

Hacker. Researching in Quantum Machine Learning in academia and in industry. Privacy enthusiast, expertise in cybersecurity. Musician.

- **Blog**
- **Quantum Machine Learning Notes**
- **Cypherpunk**
- **About**
- **Music**

Follow me:

Blog Posts

19 Jul 2018 » Selected articles on Quantum Machine Learning
18 Jul 2018 » Quantum Frobenius Distance Classifier
02 Jul 2018 » Iordanis Kerenidis' talk on quantum machine learning
16 Jun 2018 » Quantum Slow Feature Analysis, a quantum algorithm for dimensionality reduction
10 Jun 2018 » How to evaluate a classifier
15 Apr 2018 » Gather Statistics For Your Qram
15 Apr 2018 » Failed Attempt To Reverse Swap Test
18 Feb 2018 » Hamiltonian Simulation
03 Feb 2018 » Storing Data In A Quantum Computer
29 Jan 2018 » Swap Test For Distances
27 Dec 2017 » Space Estimation Of Hhl
04 Dec 2017 » Rewriting Swap Test
21 Nov 2017 » The Hhl Algorithm
06 Jan 2017 » Transavia is not recommended for travelling musicians
06 Jan 2017 » My i3 configuration for Qubes-OS
21 Aug 2016 » Migrations and functors
13 Jun 2016 » A primer on Projective Simulation: a (quantum) ML algorithm.
11 Apr 2016 » CCNOT on a Feyman's quantum computer
01 Apr 2016 » Palindromic Fibonacci in python
21 Mar 2016 » The twelvefold pythonic way (WIP post)
10 Mar 2016 » When ugly code must be written IRL: i.e. putting password inside the code like a pr0
09 Mar 2016 » Whonix AppVM won't connect to Tor after hybernate
26 Feb 2016 » Some ideas: (part two)

Model checking with Büchi automata

Intro

Theorem (S1S decidability)

Monadic second order logic is decidable.

Formal software verification is done using languages of S1S logic.

Intro

Theorem (S1S decidability)

Monadic second order logic is decidable.

Formal software verification is done using languages of S1S logic.

Definition (Büchi automata)

A Büchi automata is a tuple $A = (Q, A, \Delta, q_0, F)$, where Q is a finite set of states, A is a finite alphabet of symbols, $q_0 \in Q$ is an initial state, $F \subseteq Q$ is a set of final states, and $\delta \subseteq Q \times A \times Q$ is the transition function. A computation of A on a ω-word α is a ω-word σ on Q such that $\sigma(0) = q_0$ and, for each $i \geq 0$, $\sigma(i), \alpha(i), \sigma(i+1) \in \Delta$. A computation σ has success if $In(\sigma) \cap F = \emptyset$. An automata A accept a ω-word α if exists a successfully computation of A on α. The language $L(A)$ accepted by A is the language of ω-words.

A language is ω-regular if and only if is accepted by a Büchi automata. ω-regular languages satisfy many properties of regular languages.

Theorem (Closure of ω-languages)

If $L_1, L_2 \subseteq A^\omega$ are ω-regular, also $L_1 \cap L_2$ and $L_1 \cup L_2$ and $\bar{L_1}$ are ω-regular.

The emptiness problem for a language L given a grammar G is to decide weather $L(G) = \emptyset$.

Theorem (Decidibility of the emptiness problem)

The emptiness problem for Büchi automata is decidable

Proof.

From the definition of accepting condition for a Büchi automata, it follows that a Büchi automata accept a word if and only if it's transition graph has a cycle with a finial state reachable from the initial state. Therefore it is sufficient to find a cycle in the graph associated with the Büchi automata from an initial state to any final state. □

Classically, MST are found using BFS or DFS in time $O(m+n)$.

Theorem

The problem "FIND d SMALLEST VALUES OF DIFFERENT TYPE" (among N elements) has bounded error quantum query complexity of $O(\sqrt{Nd})$.[DHHM06]

Long story short..[DHHM06]

$$m + n > \sqrt{mn \log n}$$

#NiceToHave

Code

Data on real world graphs

Quantum key distribution

- Classical cryptography relies on computational assumption. i.e. we believe there are portables that are difficult to solve by classical computer (i.e. factoring)

- Post-quantum cryptography is CLASSICAL cryptography built on top of problem that we believe to be hard to solve by QUANTUM computer.

- Quantum cryptography is the art of using quantum communication protocols in order to build encryption schemes that are **information-theoretically secure**.

- We remove a weak link in the chain-of-trust (assuming secure hardware and implementation)

- Classical cryptography relies on computational assumption. i.e. we believe there are portables that are difficult to solve by classical computer (i.e. factoring)

- Post-quantum cryptography is CLASSICAL cryptography built on top of problem that we believe to be hard to solve by QUANTUM computer.

- Quantum cryptography is the art of using quantum communication protocols in order to build encryption schemes that are **information-theoretically secure**.

- We remove a weak link in the chain-of-trust (assuming secure hardware and implementation)

- Classical cryptography relies on computational assumption. i.e. we believe there are portables that are difficult to solve by classical computer (i.e. factoring)

- Post-quantum cryptography is CLASSICAL cryptography built on top of problem that we believe to be hard to solve by QUANTUM computer.

- Quantum cryptography is the art of using quantum communication protocols in order to build encryption schemes that are **information-theoretically secure**.

- We remove a weak link in the chain-of-trust (assuming secure hardware and implementation)

- Classical cryptography relies on computational assumption. i.e. we believe there are portables that are difficult to solve by classical computer (i.e. factoring)

- Post-quantum cryptography is CLASSICAL cryptography built on top of problem that we believe to be hard to solve by QUANTUM computer.

- Quantum cryptography is the art of using quantum communication protocols in order to build encryption schemes that are **information-theoretically secure**.

- We remove a weak link in the chain-of-trust (assuming secure hardware and implementation)

BB84 protocol

- **Classical channel**: public, vulnerable, authenticated
- **Quantum channel**: public, vulnerable. (air, fiber,...)
- Length of the key N, requires $n = (4 + \delta)N$ qubits
- Alice generates two binary strings n: $\vec{a} = a_0, \cdots, a_n$, $\vec{b} = b_0, \cdots, b_n$
- If b_i is 0 then Alice sends in the quantum channel $|a_i\rangle$, otherwise she sends $H|b_i\rangle$.
- Bob generates a random string $\vec{c}$ of length n as well. If c_i is 0 the measure, otherwise he applies H and then measure.
- **Then** they use the classical authenticated channel to reveal $\vec{b}$ and $\vec{c}$.

BB84 protocol

- **Classical channel**: public, vulnerable, authenticated
- **Quantum channel**: public, vulnerable. (air, fiber,...)
- Length of the key N, requires $n = (4 + \delta)N$ qubits
- Alice generates two binary strings n: $\vec{a} = a_0, \cdots, a_n$, $\vec{b} = b_0, \cdots, b_n$
- If b_i is 0 then Alice sends in the quantum channel $|a_i\rangle$, otherwise she sends $H|b_i\rangle$.
- Bob generates a random string $\vec{c}$ of length n as well. If c_i is 0 the measure, otherwise he applies H and then measure.
- **Then** they use the classical authenticated channel to reveal $\vec{b}$ and $\vec{c}$.

Conclusion on QKD

- By adding information-theoretic security we remove (forever) a possible weak node in the chain of trust of our infrastructure.

- Sure, we (temporarily) add the possibility of physical attacks on the hardware infrastructure...

- But the overall consequence is that we increase the cost of an attack!

Conclusion on QKD

- By adding information-theoretic security we remove (forever) a possible weak node in the chain of trust of our infrastructure.

- Sure, we (temporarily) add the possibility of physical attacks on the hardware infrastructure...

- But the overall consequence is that we increase the cost of an attack!

Conclusion on QKD

- By adding information-theoretic security we remove (forever) a possible weak node in the chain of trust of our infrastructure.

- Sure, we (temporarily) add the possibility of physical attacks on the hardware infrastructure...

- But the overall consequence is that we increase the cost of an attack!

Blindness and verification

Quantum computers give the possibility to delegate computation “in the cloud”.

Position based quantum cryptography

- The goal of position-based cryptography is for an honest party to use her spatio-temporal position as her only credentials in a cryptographic protocol.

- Position verification aims at verifying that a certain party (the prover), holds a given position in space-time.

- Example: Pizza ordering problem.

- Classically, information-theoretic security could never be obtained: it is always possible for a coalition of adversaries to convince the verifiers that the adversary are in the right position.

Position based quantum cryptography

- The goal of position-based cryptography is for an honest party to use her spatio-temporal position as her only credentials in a cryptographic protocol.

- Position verification aims at verifying that a certain party (the prover), holds a given position in space-time.

- Example: Pizza ordering problem.

- Classically, information-theoretic security could never be obtained: it is always possible for a coalition of adversaries to convince the verifiers that the adversary are in the right position.

Position based quantum cryptography

- The goal of position-based cryptography is for an honest party to use her spatio-temporal position as her only credentials in a cryptographic protocol.

- Position verification aims at verifying that a certain party (the prover), holds a given position in space-time.

- Example: Pizza ordering problem.

- Classically, information-theoretic security could never be obtained: it is always possible for a coalition of adversaries to convince the verifiers that the adversary are in the right position.

Position based quantum cryptography

- The goal of position-based cryptography is for an honest party to use her spatio-temporal position as her only credentials in a cryptographic protocol.

- Position verification aims at verifying that a certain party (the prover), holds a given position in space-time.

- Example: Pizza ordering problem.

- Classically, information-theoretic security could never be obtained: it is always possible for a coalition of adversaries to convince the verifiers that the adversary are in the right position.

- Currently, a class of attacks requires exponential entanglement shared between the attackers to succeed.

- Open Problem: Prove that such attack is optimal will give a secure protocol for quantum position based cryptography.

Why is cool?

- We unlock a possibility that previously was not allowed by classical cryptography!

- (like cryptocurrencies) We don't know yet HOW MUCH this will be useful.

TESTING QUANTUM SOFTWARE. HELP!

Thanks for your time,
there is never enough. (cit.)

`https://luongo.pro`
Twitter: @scinawa

Divesh Aggarwal, Gavin K Brennen, Troy Lee, Miklos Santha, and Marco Tomamichel.

Quantum attacks on bitcoin, and how to protect against them.
arXiv preprint arXiv:1710.10377, 2017.

Gilles Brassard, Peter Hoyer, Michele Mosca, and Alain Tapp.

Quantum amplitude amplification and estimation.
Contemporary Mathematics, 305:53–74, 2002.

Mathieu Bozzio, Adeline Orieux, Luis Trigo Vidarte, Isabelle Zaquine, Iordanis Kerenidis, and Eleni Diamanti.

Experimental investigation of practical unforgeable quantum money.
npj Quantum Information, 4(1):5, 2018.

Shantanav Chakraborty, András Gilyén, and Stacey Jeffery.

The power of block-encoded matrix powers: improved regression techniques via faster hamiltonian simulation.
arXiv preprint arXiv:1804.01973, 2018.

Christoph Dürr, Mark Heiligman, Peter HOyer, and Mehdi Mhalla.

Quantum query complexity of some graph problems.
SIAM Journal on Computing, 35(6):1310–1328, 2006.

András Gilyén, Yuan Su, Guang Hao Low, and Nathan Wiebe.

Quantum singular value transformation and beyond: exponential improvements for quantum matrix arithmetics.
arXiv preprint arXiv:1806.01838, 2018.

Iordanis Kerenidis and Alessandro Luongo.

Quantum classification of the MNIST dataset via Slow Feature Analysis.
arXiv preprint arXiv:1805.08837, 2018.

Leslie Lamport.

Proving the correctness of multiprocess programs.
IEEE transactions on software engineering, (2):125–143, 1977.

Len Sassaman, Meredith L Patterson, Sergey Bratus, and Michael E Locasto.

Security applications of formal language theory.