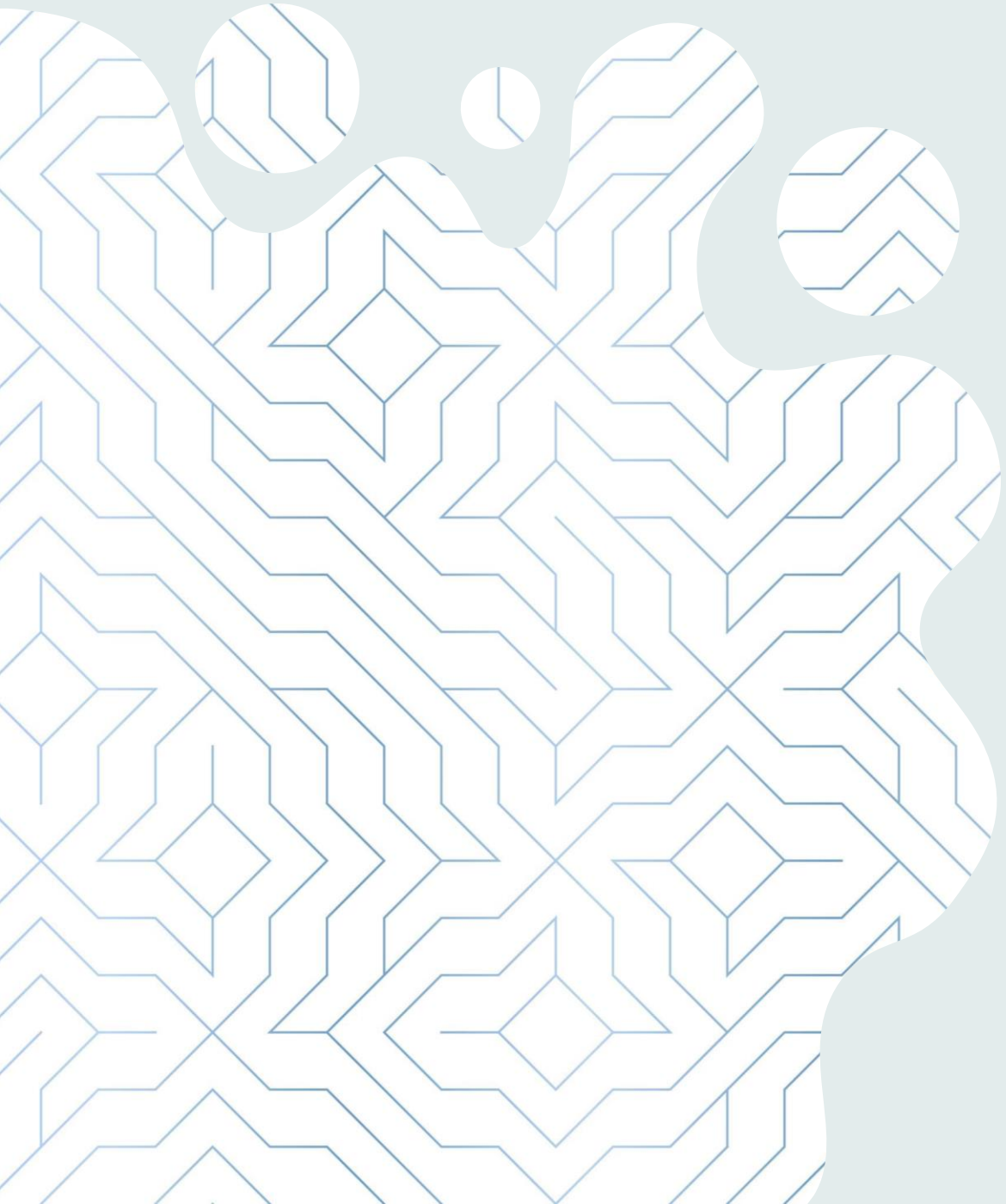


Kvantové korelace ve službách kryptografie

Adam Brůna, Lukáš Balon, Richard Daniel Maštovský, Jakub Vojtek

Vernamova šifra

" "	0	00000
"A"	1	00001
"B"	2	00010
"C"	3	00011
"D"	4	00100
"E"	5	00101

Ahoj -> 1 8 15 10 -> 00001 01000 01111 01010

Vernamova šifra

Původní zpráva	0	0	0	1
Tajný klíč	1	1	0	1
Zašifrovaná zpráva	1	1	0	0

Původní zpráva	0	0	0	1

Bezpečnostní předpoklady Vernamovy šifry

- Náhodný kód
- Maximálně jedno použití
- Soukromý klíč musí být stejně dlouhý jako zpráva
- **Potřeba udržet kód v tajnosti**

Kvantová mechanika

- qubit
- Systémy se dvěma stavy $|0\rangle$ a $|1\rangle$
  - Polarizace fotonů
  - Spin částic
  - 2 supravodivé stavy (IBMQ)
  - ...
- Měření mění stav qubitu

Superpozice

- Systém se může nacházet ve stavech $|0\rangle$ a $|1\rangle$, ale i v libovolné superpozici

$$|\Psi\rangle = \alpha|0\rangle + \beta|1\rangle$$

Stav $|0\rangle$ naměříme s pravděpodobností $|\alpha|^2$

$$|\alpha|^2 + |\beta|^2 = 1$$

Měření v různých bázích

- Například v bázi +/-

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

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

- Dosazením za $|0\rangle$ a $|1\rangle$ dostaneme

$$|\Psi\rangle = \alpha|0\rangle + \beta|1\rangle = \alpha_1|+\rangle + \beta_1|-\rangle$$

Stav $|+\rangle$ naměříme s pravděpodobností $|\alpha_1|^2$

2 qubity

- Obecný stav dvou qubitů je superpozice stavů |00⟩ |01⟩ |10⟩ a |11⟩

$$|\Psi\rangle = \alpha_1|00\rangle + \alpha_2|01\rangle + \alpha_3|10\rangle + \alpha_4|11\rangle$$

- Pro kvantovou kryptografii je důležitý stav maximální provázanosti

$$|e\rangle = \frac{1}{\sqrt{2}}|00\rangle + \frac{1}{\sqrt{2}}|11\rangle$$

Maximální kvantové provázání

- Výsledky měření na jednotlivých qubitech v bázi 0/1 i +/- jsou na sobě závislé
- Výsledek měření se určí změřením jednoho z qubitů

- Lze měření na provázaných qubitech použít k přenosu informace?

Operace s qubity

- X/NOT
  - Zaměnuje 1 a 0
- Hadamardova brána
  - Mění $|0\rangle$ na $|+\rangle$ a $|1\rangle$ na $|-\rangle$ a obráceně
- CNOT – 2 qubity $|i, j\rangle$
  - Mění j na i + j mod 2

$|00\rangle \rightarrow |00\rangle$
$|01\rangle \rightarrow |01\rangle$
$|10\rangle \rightarrow |11\rangle$
$|11\rangle \rightarrow |10\rangle$

Kvantové šifrování

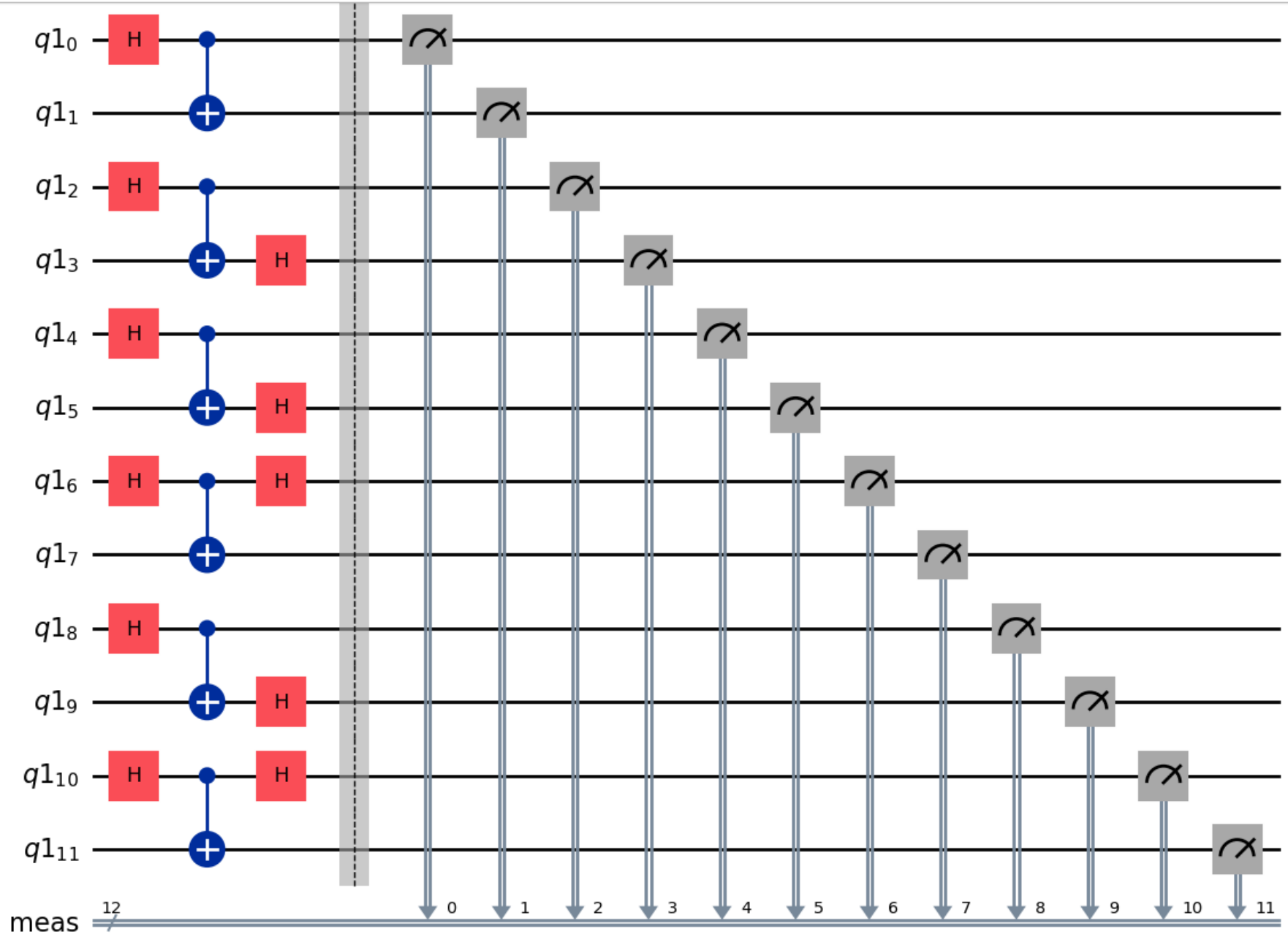
- Alice a Bob
- Zpráva o n znacích
- Vytvoří 2n maximálně provázaných párů
  - Hadamardova brána a CNOT
  - Provázané qubity
  - Alice a Bob dostanou po jednom qubitu z každého provázaného páru

Kvantové šifrování

- A i B měří každý qubit náhodně každý buď v bázi 0/1, nebo +/-
  - Bázi +/- převedou na 01 Hadamardovou bránou
- Dále zveřejní, kdy měřili jakou bází
- Shoda báze = shoda výsledku => tajný klíč

Launcher | QKD.ipynb

Code | Notebook | Python 3 (ipykernel)

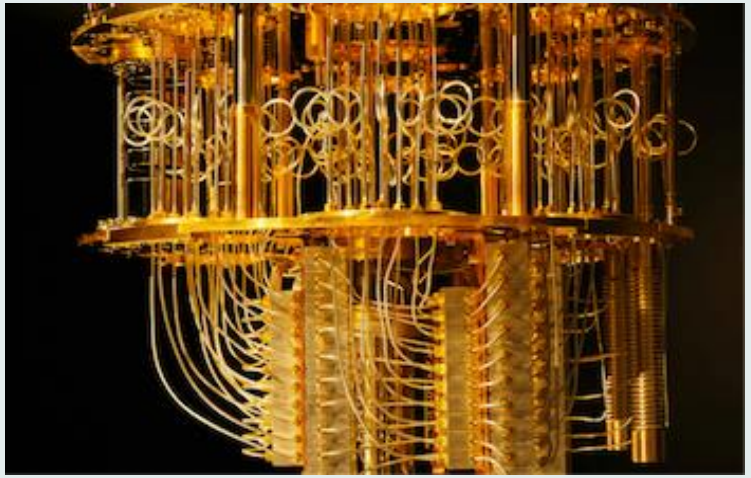


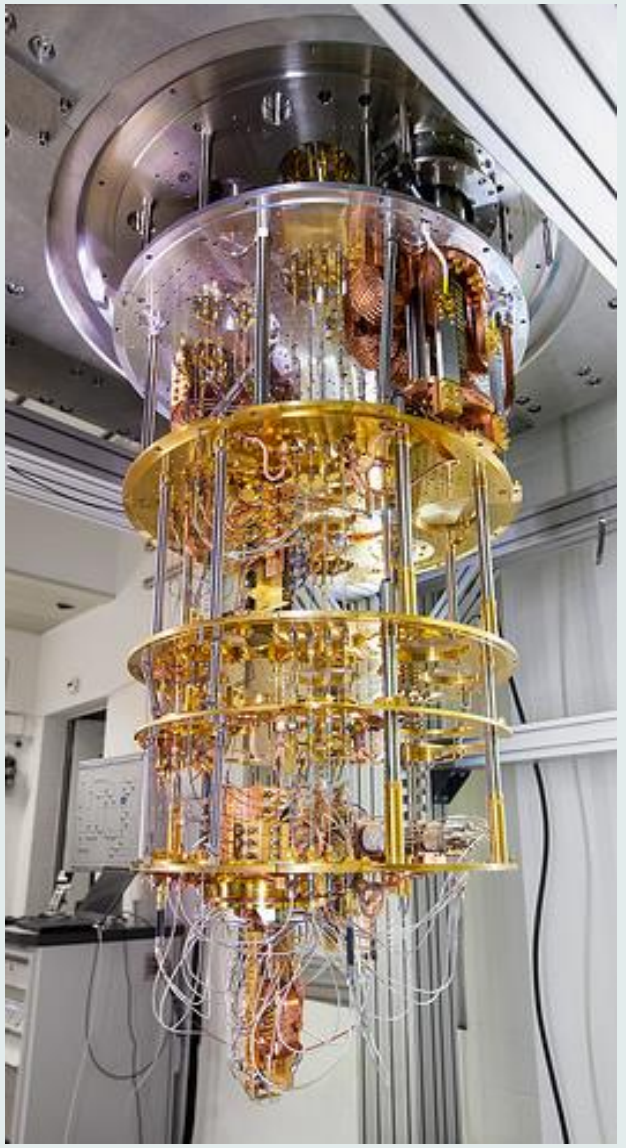
Výhody

- Náhodnost tajného klíče
- Bezpečnost tajného klíče
  - Provázanost qubitů lze ověřit
- Lze odhalit útočníka

IBMQ

- Síť kvantových počítačů
- Využita námi k simulaci
- Velká chybovost dnešních modelů





Launcher | QKD.ipynb

Code | Notebook | Python 3 (ipykernel)

[11]:
```
secretMessage = 'ahoj ahoj ahoj'

intendedKeyLength = 5*len(secretMessage)
# numberOfQubitsPerParty = math.floor(processorSizeSimulator/2)
numberOfQubitsPerParty = math.floor(processorSizeQPU/2)

bitStringAlice = ''.join(random.choice('01') for i in range(0,2*intendedKeyLength))
bitStringBob = ''.join(random.choice('01') for i in range(0,2*intendedKeyLength))
bitStringBob = bitStringAlice


splitBitStringAlice = [bitStringAlice[i:i+numberOfQubitsPerParty] for i in range(0,len(bitStringAlice),numberOfQubitsPerParty)]
splitBitStringBob = [bitStringBob[i:i+numberOfQubitsPerParty] for i in range(0,len(bitStringBob),numberOfQubitsPerParty)]

circuits = [createCircuit(splitBitStringAlice[i],splitBitStringBob[i]) for i in range(0,len(splitBitStringAlice))]

jobIdArray = qpuRunCircuits(circuits)
# print(jobIdArray)
```

```
qiskit_runtime_service.check_pending_jobs:WARNING:2025-06-24 10:35:00,454: The pending jobs limit has been reached. Waiting for job <RuntimeJob('d1d66c1v3z50008am29g', 'sampler')> to finish before submitting the ne
xt one.
qiskit_runtime_service.check_pending_jobs:WARNING:2025-06-24 10:35:10,467: The pending jobs limit has been reached. Waiting for job <RuntimeJob('d1d66ehn2txg008eez2g', 'sampler')> to finish before submitting the ne
xt one.
qiskit_runtime_service.check_pending_jobs:WARNING:2025-06-24 10:35:24,413: The pending jobs limit has been reached. Waiting for job <RuntimeJob('d1d66j2qf56g00804zag', 'sampler')> to finish before submitting the ne
xt one.
qiskit_runtime_service.check_pending_jobs:WARNING:2025-06-24 10:35:29,834: The pending jobs limit has been reached. Waiting for job <RuntimeJob('d1d66jjmya70008nm4v0', 'sampler')> to finish before submitting the ne
xt one.
Submitted to IBMQ: 2025-06-24 10:35:33.557681
Backend: <IBMBackend('ibm_brisbane')>
Batch id: d1d66ahn2txg008eez1g
JobIdArray: ['d1d66b93grvg008n3vng', 'd1d66c1v3z50008am29g', 'd1d66chqf56g00804z6g', 'd1d66e1mya70008nm4r0', 'd1d66ehn2txg008eez2g', 'd1d66f1qf56g00804z80', 'd1d66gaqf56g00804z90', 'd1d66gtqf56g00804z9g', 'd1d66hjm
ya70008nm4tg', 'd1d66j2qf56g00804zag', 'd1d66jjmya70008nm4v0', 'd1d66kjmya70008nm4w0', 'd1d66m2mya70008nm4wg', 'd1d66mtn2txg008eez5g']
```

[12]:
```
result = retrieveResultsQPU(jobIdArray)

measuredValuesAlice, measuredValuesBob = processMeasurementResult(result)
keyAlice, keyBob = selectKeys(bitStringAlice, bitStringBob, measuredValuesAlice, measuredValuesBob)

encodedMessage = encode(stringToBinary(secretMessage.lower()),keyAlice)
decodedMessage = binaryToString(decode(encodedMessage,keyBob))


print(secretMessage)
print()
print(decodedMessage)
```

```
ahoj ahoj ahoj

oh mm j cuekte
```

Zdroje Obrázků

https://lille-ibmq.sciencesconf.org/index7819.html?forward-action=index&forward-controller=index&lang=en

https://www.i-programmer.info/news/204-challenges/11436-ibm-announces-quantum-computing-prizes.html

Děkujeme za pozornost