Quantum Computing en Security

Magnum opus

Opleiding: Systeem- en Netwerkbeheer

Academiejaar: 2024-2025

Jelle De Souter

Voornaam en Naam promotor/ eindwerkbegeleider/ eindprojectbegeleiders

Inhoud

De impact van Quantum Computing op Security

1 Inleiding

In dit inleidende hoofdstuk verken ik de fundamenten van quantum computing en de groeiende relevantie ervan op het gebied van cybersecurity. Ik leg de basisprincipes van de technologie uit, waarom deze de moderne cryptografie kan verstoren en wat de motivatie was voor de keuze van dit onderwerp binnen de context van systeem- en netwerkbeheer.

1.1 Wat is quantum Computing?

Quantum computing is een opkomende technologie in de computerwetenschap die gebruik maakt van de principes van kwantummechanica voor het uitvoeren van berekeningen. Deze manier van denken is veel anders dan die van klassieke computers. Terwijl klassieke computers gebruik maken van bits (0 of 1), gebruiken quantum computers qubits. Deze qubits kunnen beide 0 en 1 tegelijkertijd zijn door een eigenschap genaamd superpositie en verstrengeling. Dit betekent dat de staat van één qubit afhangt van de staat van een andere qubit, het maakt niet uit hoe ver deze van elkaar zijn verwijderd. Dit maakt het mogelijk voor quantum computers om complexe berekeningen in minder stappen te verwerken, waardoor ze exponentieel sneller worden bij het oplossen van bepaalde problemen, inclusief deze waarop huidige encryptie gebaseerd is.

1.2 Waarom is dit relevant voor IT/security

Quantum computing is zeer relevant voor cybersecurity omdat veel van de huidige cryptografische systemen die onze digitale infrastructuur beveiligen afhankelijk zijn van de moeilijkheidsgraad van bepaalde wiskundige problemen, zoals het ontbinden van grote priemgetallen of het berekenen van elliptische krommen. Quantum computers die gebruik maken van algoritmes zoals Shor's, kunnen deze problemen significant sneller oplossen dan klassieke computers. Als quantum computers in de praktijk toepasbaar worden, zouden veelgebruikte versleutelingsmethoden zoals

RSA en ECC gekraakt kunnen worden. Dit zou de vertrouwelijkheid van gegevens en de integriteit van communicatie op het internet in gevaar kunnen brengen.

1.3 Wat motiveerde mij om dit onderwerp te kiezen

Quantum computing trok mijn aandacht omdat het alles kan veranderen wat we nu kennen over beveiliging. Wat me vooral opvalt is dat dit niet iets is voor de verre toekomst - experts denken dat quantum computers onze huidige encryptie kunnen breken binnen 10-15 jaar. Dat betekent dat we als toekomstige IT'ers nu al moeten nadenken over welke maatregelen we moeten nemen.

Wat me ook interesseert is dat hackers nu al versleutelde data kunnen verzamelen om het later te ontcijferen wanneer quantum computers vrij beschikbaar zijn. Dit heet "harvest now, decrypt later" en het laat zien dat we echt geen tijd te verliezen hebben.

Door dit onderwerp te kiezen wil ik meer leren over quantum computing en uitzoeken wat dit praktisch betekent voor ons vakgebied. Mijn doel is om een duidelijk beeld te geven van de gevaren en concrete acties die we nu al kunnen ondernemen om ons voor te bereiden.

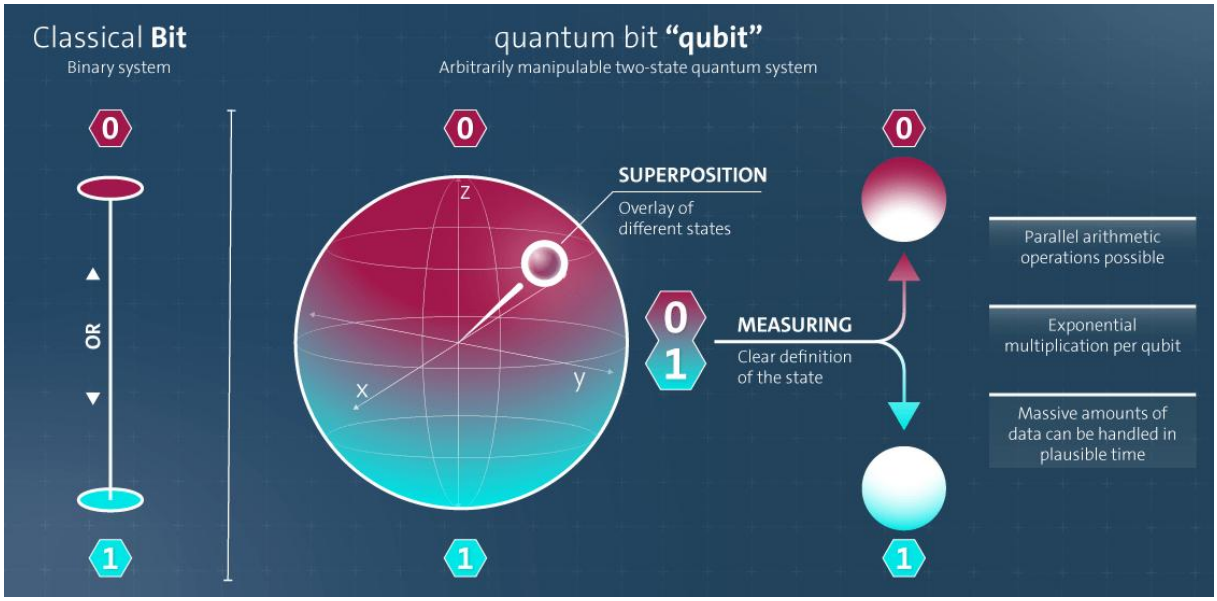
2 Wat is Quantum Computing?

Dit hoofdstuk biedt een technisch overzicht van quantum computing. Ik leg uit hoe qubits functioneren volgens principes zoals superpositie en verstrengeling, en hoe deze concepten fundamenteel verschillen van klassieke computing. Door deze verschillen te benadrukken, kunnen we beter begrijpen waarom quantumcomputers zo krachtig zijn, en zulke gevaren vormen voor de huidige cryptografie.

2.1 Korte technische uitleg: qubits, superpositie, verstrengeling

Quantum computers zijn gebaseerd op qubits, die in tegenstelling tot binaire bits tegelijkertijd in een combinatie van 0 en 1 kunnen bestaan. Dit fenomeen staat bekend als superpositie. Een ander belangrijk principe is verstrengeling

(entanglement), waarbij twee of meer qubits met elkaar worden verbonden, zodat de toestand van de ene qubit direct de toestand van de andere beïnvloedt, zelfs over lange afstanden. Deze eigenschappen maken het voor quantum computers mogelijk om grote hoeveelheden datasets en complexe algoritmes sneller en meer efficiënt te verwerken.



Bron: https://devopedia.org/qubit

2.2 Verschillen met klassieke computing

Klassieke computers gebruiken transistors en logische poorten voor het verwerken van informatie in binair formaat. Elke operatie wordt stap-voor-stap behandeld. Quantum computers, daarentegen, opereren met quantum gates, wat parallelle berekeningen door superpositie en verstrengeling mogelijk maakt. Dit maakt exponentiële schaalbaarheid mogelijk, wat gebieden zoals cryptografie, medicijnontdekking en kunstmatige intelligentie zou kunnen transformeren. Quantum computing brengt echter ook uitdagingen met zich mee, zoals foutcorrectie en qubit-stabiliteit.

3 Quantum Computing en Cryptografie

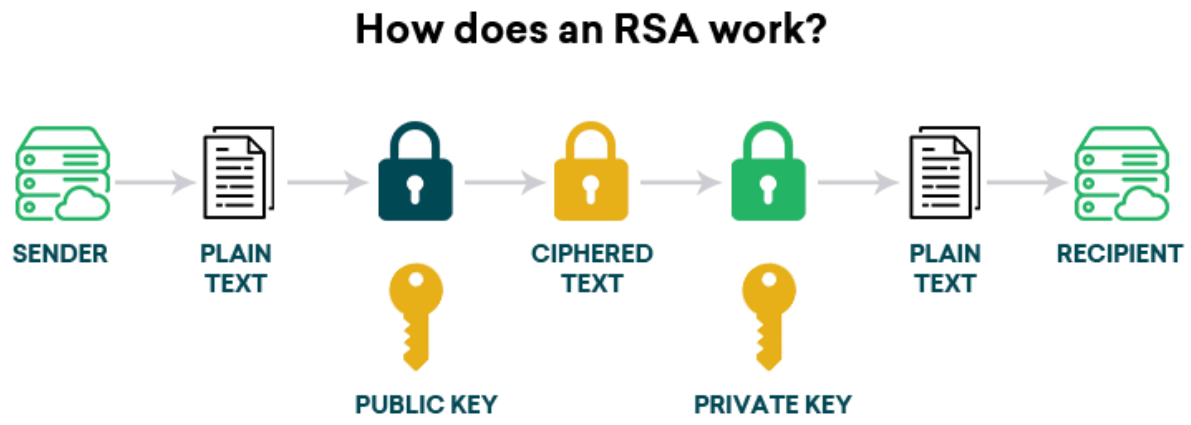
In dit hoofdstuk onderzoek ik de verbanden tussen quantum computing en klassieke cryptografie. Hoe huidige encryptiemethoden werken en waarom quantum computers

deze kunnen breken. Het meeste aandacht zal ik aan Shor's algoritme besteden, het meest bekende quantum algoritme dat in staat is om grote getallen efficiënt te ontbinden, en vormt daarmee een bedreiging voor veelgebruikte systemen als RSA en ECC.

3.1 Hoe werkt klassieke cryptografie (bijv. RSA, ECC)?

Klassieke cryptografie is gebouwd op wiskundige problemen die makkelijk te berekenen zijn in één richting maar extreem moeilijk om terug te draaien zonder specifieke kennis. RSA, één van de meest wijdverspreide encryptie-algoritmes, is gebaseerd op de moeilijkheid om grote priemgetallen te ontbinden. ECC (Elliptic Curve Cryptography) gebruikt de moeilijkheid van het elliptic curve discrete logarithm problem. Deze systemen vormen de basis van beveiligde communicatie, zoals HTTPS, VPN's en digitale handtekeningen.

Bijvoorbeeld, RSA genereert een publieke en private sleutelpaar. De publieke sleutel wordt gebruikt om het bericht te versleutelen en de private sleutel om deze te ontsleutelen. De veiligheid schuilt in het feit dat het vermenigvuldigen van twee grote priemgetallen eenvoudig is, maar het factoren van het resultaat naar die priemgetallen met klassieke berekeningen vrijwel onmogelijk is. Dat is een taak die duizenden jaren kan duren.



Bron: https://www.clickssl.net/blog/what-is-rsa

3.2 Waarom vormen quantumcomputers hier een bedreiging voor?

Quantum computers vormen een serieus gevaar voor klassieke cryptografie vanwege hun vermogen om problemen op te lossen die voor traditionele systemen als onmogelijk worden beschouwd. Algoritmes zoals dat van Shor kunnen grote getallen in polynomiale tijd ontbinden met behulp van de principes van de kwantummechanica. Dit betekent dat quantum computers, als ze eenmaal krachtig genoeg zijn, RSA-encryptie binnen enkele minuten of zelfs seconden kunnen kraken, waardoor een groot deel van de huidige digitale beveiliging overbodig wordt.

3.3 Wat is Shor's algoritme?

Shor's algoritme, ontwikkeld door wiskundige Peter Shor in 1994, is een quantum algoritme dat in staat is om gehele getallen efficiënt te ontbinden. Het maakt gebruik van quantumparallellisme en periodiciteit om niet-triviale delers van een samengesteld getal te vinden. Dit vormt een directe bedreiging voor RSA, dat afhankelijk is van de moeilijkheidsgraad van deze factorisatie. Het algoritme werkt door het factorisatieprobleem te transformeren naar een periodebepalend probleem, dat snel kan worden opgelost op een quantumcomputer.

Een vereenvoudigd voorbeeld: gegeven een getal N = 15, kan Shor's algoritme efficiënt vaststellen dat 3 en 5 de niet-triviale factoren zijn. Hoewel 15 op elke computer triviaal te ontbinden is, gebruiken RSA-sleutels doorgaans getallen met meer dan 600 decimalen. Dit is onmogelijk om met klassieke technieken in een praktische hoeveelheid tijd te ontbinden, maar wel haalbaar voor toekomstige quantummachines.

4 Wat betekent dit voor netwerkbeveiliging?

De overgang naar het quantumtijdperk brengt aanzienlijke risico's met zich mee voor bestaande netwerkinfrastructuren, vooral voor die welke afhankelijk zijn van klassieke cryptografische technieken. Als quantumcomputers krachtig genoeg worden, kunnen ze veelgebruikte cryptografische protocollen met openbare sleutels, zoals RSA, DSA en ECC, kraken.

Deze dreiging creëert een situatie die bekendstaat als "harvest now, decrypt later". Kwaadwillenden onderscheppen en slaan versleutelde communicatie mogelijk al op met als doel deze te ontsleutelen zodra quantumcomputers beschikbaar zijn. Dit is gevaarlijk voor gevoelige gegevens met lange vertrouwelijkheidsvereisten - denk aan medische dossiers, overheidsdocumenten en bedrijfsgeheimen.

Netwerkprotocollen zoals TLS (Transport Layer Security), IPsec en SSH vertrouwen allemaal op cryptografie met openbare sleutels voor authenticatie en sleuteluitwisseling. Als deze mechanismen falen, kunnen aanvallers zich voordoen als gebruikers, vertrouwelijke gegevens onderscheppen of schadelijke content in communicatiestromen injecteren.

Bovendien kunnen digitale handtekeningen die worden gebruikt om software, updates en certificaten te verifiëren, worden vervalst, wat leidt tot wijdverbreide schending van het vertrouwen op internet.

4.1 Wat zijn de risico's voor huidige systemen?

Veel systemen die tegenwoordig worden gebruikt, vertrouwen voor kritieke functies op cryptografie met openbare sleutels. De opkomst van quantum computing betekent dat organisaties in de nabije toekomst kwetsbaar kunnen zijn als ze nu geen actie ondernemen. Versleuteld verkeer kan met terugwerkende kracht worden ontsleuteld, kritieke communicatie kan worden blootgelegd en sleutelauthenticatiesystemen kunnen worden ondermijnd.

4.2 Wanneer wordt dit een echt probleem?

Deskundigen verwachten dat quantumcomputers die RSA-2048 kunnen kraken tussen 2030 en 2040 beschikbaar zullen zijn. Maar de migratie naar quantumveilige systemen kan wel tien jaar of langer duren, vooral voor grote infrastructuren. Dit betekent dat we ons al in het kritieke venster bevinden om de transitie te starten.

Overheden en instanties zoals NIST en NSA adviseren om nu te beginnen met hybride benaderingen en systemen voor te bereiden op post-quantum algoritmes.

4.3 Zijn er reeds incidenten of proof-of-concepts?

Hoewel er nog geen echte cryptografische ontcijfering door een quantumcomputer heeft plaatsgevonden, tonen verschillende proof-of-concept-experimenten de theoretische dreiging aan. Voorbeelden hiervan zijn kleinschalige implementaties van Shor's algoritme op quantumprocessoren en ontwikkelingen in quantumhardware, zoals Google's Sycamore die quantumsuperioriteit heeft bereikt.

Daarnaast worden er al pilots van Quantum Key Distribution (QKD)-netwerken uitgerold in landen als China en Nederland, wat de reële interesse in quantumveilige communicatie aantoont.

5 Post-Quantum Cryptografie

In dit hoofdstuk onderzoek ik hoe de cybersecuritywereld zich voorbereidt op de quantumtoekomst. Post-quantumcryptografie (PQC) richt zich op de ontwikkeling van nieuwe cryptografische algoritmes die zelfs tegen quantumcomputers veilig zijn. Ik bekijk wat PQC inhoudt, welke organisaties de ontwikkeling ervan leiden en welke algoritmes de meest veelbelovende oplossingen lijken te zijn.

5.1 Wat is het?

Post-quantumcryptografie verwijst naar cryptografische algoritmes die op klassieke computers kunnen draaien, maar die ontworpen zijn om aanvallen van zowel klassieke als quantumsystemen te weerstaan. Deze algoritmes zijn speciaal ontworpen om zich te verdedigen tegen bedreigingen van quantumalgoritmes zoals die van Shor en Grover.

In tegenstelling tot quantumcryptografie (bijv. Quantum Key Distribution) vereist PQC geen quantumhardware. Het gebruikt moeilijke wiskundige problemen die, gebaseerd op de huidige inzichten, zelfs voor quantumcomputers moeilijk op te lossen zijn.

Het doel is om een direct inzetbare vervanging te ontwikkelen voor huidige algoritmes zoals RSA en ECC, wat betekent dat we bestaande systemen kunnen updaten zonder nieuwe infrastructuur of hardware nodig te hebben.

5.2 Wat doen organisaties zoals NIST eraan?

Het National Institute of Standards and Technology (NIST) in de Verenigde Staten leidt een wereldwijd initiatief om post-quantum cryptografische algoritmes te standaardiseren. In 2016 lanceerde NIST een openbare competitie om quantumresistente algoritmes te evalueren en te selecteren.

Dit proces verloopt in meerdere ronden. In juli 2022 maakte NIST bekend dat vier algoritmes geselecteerd waren voor standaardisatie, terwijl andere nog in evaluatie zijn.

Geselecteerde algoritmen door NIST (per 2022):

– **CRYSTALS-Kyber:** Sleutelvaststelling (vervanging voor RSA/ECC-sleuteluitwisseling)
– **CRYSTALS-Dilithium:** Digitale handtekeningen
– **FALCON:** Digitale handtekeningen
– **SPHINCS+:** Stateless hash-gebaseerde digitale handtekeningen

Deze algoritmen zijn gekozen vanwege hun sterke beveiligingsbewijzen, prestaties en compatibiliteit met bestaande protocollen.

6 Hands-on experiment: Shor’s algoritme simuleren met Qiskit

In dit hands-on experiment toon ik aan hoe Shor's algoritme werkt in de praktijk en demonstreer ik de directe bedreiging die quantumcomputers vormen voor moderne cryptografie. Ik ga één van de bekendste quantumalgoritmes implementeren: Shor's algoritme voor het factoriseren van getallen. Ik zal dit algoritme gebruiken om het getal 15 te factoriseren in zijn priemfactoren (3 en 5).

Hoewel dit op het eerste gezicht een eenvoudig voorbeeld lijkt, illustreert het de kracht van quantumcomputing. Het algoritme dat ik implementeer is hetzelfde als wat in de toekomst gebruikt kan worden om grote RSA-sleutels te kraken, alleen dan toegepast op een kleinere schaal die we kunnen simuleren op een klassieke computer. Door dit experiment uit te voeren, krijgen we inzicht in zowel de mogelijkheden als de huidige beperkingen van quantumcomputing voor cryptografische toepassingen.

Voor dit experiment gebruik ik Qiskit, een open-source quantumcomputing framework ontwikkeld door IBM. Ik zal eerst de ontwikkelomgeving opzetten, het algoritme implementeren, de resultaten analyseren, en ten slotte reflecteren op de implicaties voor cybersecurity. Ik zal alle stappen in detail uitleggen zodat je dit experiment ook zelf kunt uitproberen.

6.1 Installatie van Qiskit

Het installeren van Qiskit op Windows met VS Code vereist verschillende stappen om een goede werkomgeving te garanderen. Dit onderdeel leg het volledige installatieproces stap per stap uit.

6.1.1 benodigdheden

Voor dat je Qiskit installeert controleer dat je deze programma's/rechten hebt op uw computer:

– Python 3.8 of nieuwer
– Visual Studio Code met de Python extensie
– Windows Terminal of Command Prompt
– Administrator rechten op jouw computer

6.2 Stap-voor-stap installatie proces

6.2.1 Stap 1: Zet een virtuele python omgeving op

Het is best practice om een dedicated virtuele omgeving te maken voor quantum computing projecten om afhankelijkheidsconflicten te vermijden.

Open command prompt als Administrator en navigeer naar de projectfolder waar je het experiment zal uitvoeren:

```
C:\Users\jelle>cd "C:\Users\jelle\school-niet-onedrive\hands on>"
```

Maak de virtuele omgeving aan en activeer het:

```
C:\Users\jelle\school-niet-onedrive\hands on>python -m venv quantum_env

C:\Users\jelle\school-niet-onedrive\hands on>quantum_env\Scripts\activate

(quantum_env) C:\Users\jelle\school-niet-onedrive\hands on>
```

U ziet dat er nu (quantum_env) staat aan het begin van de command line, dit betekent dat de virtuele omgeving actief is.

6.2.2 Stap 2: Qiskit en afhankelijkheden installeren:

We gaan verder op de command line van stap 1. Nu gaan we al de Qiskit packeten installeren.

```
(quantum_env) C:\Users\jelle\school-niet-onedrive\hands on>pip install qiskit
```

```
(quantum_env) C:\Users\jelle\school-niet-onedrive\hands on>pip install qiskit-aer
```

```
(quantum_env) C:\Users\jelle\school-niet-onedrive\hands on>pip install matplotlib
```

```
(quantum_env) C:\Users\jelle\school-niet-onedrive\hands on>pip install qiskit-algorithms
```

```
(quantum_env) C:\Users\jelle\school-niet-onedrive\hands on>pip install ipykernel
```

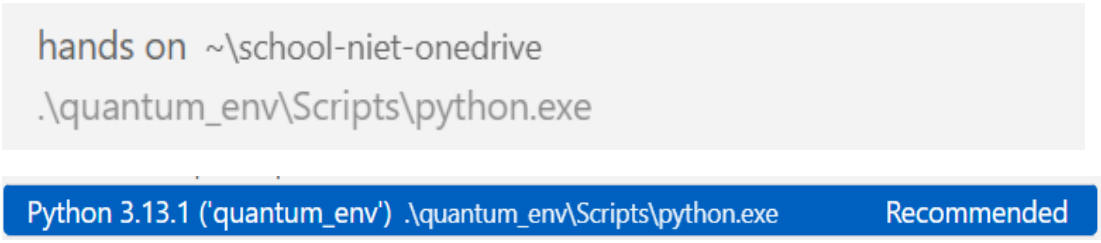
6.2.3 Stap 3: Configureren van VS code

1 Open Visual Studio Code

2 Selecteer het quantum_env als de Python interpeter

– Druk: Ctrl+Shift+P en selecteer de interpeter

– Selecteer de virtuele omgeving die u in de vorige stappen heeft aangemaakt (in mijn geval ziet dit er zo uit):



6.3 Uitvoeren van het shor algoritme

6.3.1 Korte uitleg van Shor's algoritme

Het algoritme van Shor, ontwikkeld door wiskundige Peter Shor in 1994, vormt een van de grootste bedreigingen die quantum computing vormt voor de klassieke cryptografie. In dit hoofdstuk worden de principes en werking van het algoritme uitgelegd.

6.3.2 De wiskundige basis

In de kern lost Shor's algoritme het factorisatieprobleem van gehele getallen efficiënt op. Voor een gegeven geheel getal N vindt het de priemfactoren. Hoewel dit eenvoudig klinkt, wordt dit probleem voor zeer grote getallen rekenintensief voor klassieke computers, wat de beveiligingsbasis vormt voor RSA-encryptie.

Shors algoritme maakt gebruik van twee belangrijke wiskundige inzichten:

1. **Reductie tot periodebepaling:** Het factorisatieprobleem kan worden gereduceerd tot het bepalen van de periode van een functie f(x) = a^x mod N, waarbij 'a' een willekeurig gekozen geheel getal is dat relatief priem is tot N.
2. **Quantum Fourier Transformatie:** quantumcomputers kunnen deze periode efficiënt bepalen met behulp van de quantum Fourier Transformatie (QFT), die exponentieel sneller is dan klassieke Fourier transformaties.

6.3.3 Stappen van het shor algoritme

Shor's algoritme werkt in drie eenvoudige stappen:

Stap 1: Voorbereiding (Klassieke Computer)

Denk hieraan als het voorbereiden van een puzzel:

1. **Controleer of het getal makkelijk te ontbinden is**: Is het getal N even? Dan is 2 een factor en zijn we klaar.
2. **Kies een willekeurig getal**: Kies een willekeurig getal 'a' tussen 1 en N.
3. **Snelle controle**: Bereken de grootste gemeenschappelijke deler van a en N. Als dit niet 1 is, hebben we geluk en hebben we al een factor gevonden!
4. **Als we geen geluk hebben**: Ga naar de quantum stap.

Stap 2: Quantum gedeeldte (Het Moeilijke Deel)

Hier excelleren quantum computers:

1. **Zet quantum computer klaar**: Maak qubits klaar om alle mogelijke waarden tegelijk te berekenen.
2. **Maak superpositie**: Zet alle qubits in een "quantum superpositie", ze zijn alle mogelijke waarden tegelijkertijd.
3. **Pas de functie toe**: Bereken a^x mod N voor alle waarden van x tegelijkertijd.
4. **Quantum Fourier Transformatie**: Een speciale quantum operatie die het patroon (de periode) vindt.
5. **Meet**: Meet de qubits om de periode 'r' te vinden.

Stap 3: Maak het Af (Klassieke Computer)

Gebruik het quantum resultaat om de factoren te vinden:

1. **Controleer of het bruikbaar is**: Als de periode 'r' oneven is, hebben we pech, probeer opnieuw met een ander getal 'a'.
2. **Bereken factoren**: Gebruik de formule: bereken gcd(a^(r/2) - 1, N) en gcd(a^(r/2) + 1, N).
3. **Succes**: Een van deze berekeningen geeft ons een factor van N.

Eenvoudig Voorbeeld met N = 15:

Stap 1: Kies a = 7 (willekeurig)

Stap 2: Quantum computer vindt dat 7^x mod 15 een periode heeft van r = 4

Stap 3: Bereken gcd(7^2 - 1, 15) = gcd(48, 15) = 3 ✓

Resultaat: 15 = 3 × 5

Waarom dit RSA breekt: Klassieke computers moeten veel waarden een voor een proberen. Quantum computers kunnen alle waarden tegelijkertijd proberen door superpositie te gebruiken, waardoor ze exponentieel sneller zijn.

6.4 Waarom het belangrijk is voor de beveiliging

RSA-encryptie is afhankelijk van de moeilijkheidsgraad van het ontbinden van grote getallen. Het kraken van een 2048-bits RSA-sleutel zou met klassieke algoritmes duizenden jaren duren. Met een voldoende krachtige quantumcomputer die Shor's algoritme draait, zou deze sleutel theoretisch binnen enkele uren of dagen gekraakt kunnen worden.

6.5 Simulatie van het algoritme op een klein getal (in dit geval 15)

In deze sectie implementeer ik een vereenvoudigde versie van Shor's algoritme om het getal 15 te ontbinden met behulp van het bijgewerkte Qiskit-framework. Dit dient als een praktische demonstratie van hoe quantumcomputing factorisatieproblemen kan oplossen.

6.5.1 Waarom 15?

Het getal 15 is ideaal om Shors algoritme te demonstreren omdat:

- Het klein genoeg is om te ontbinden op kwantumsimulatoren met een beperkt aantal qubits.
- De factoren (3 en 5) zijn beide priemgetallen.
- Het groot genoeg is om de principes van het algoritme te demonstreren.

6.5.2 De python code

```python
# shors_algorithm_updated.py
# Import necessary libraries for updated Qiskit
from qiskit import QuantumCircuit, transpile
from qiskit_aer import Aer
from qiskit.visualization import plot_histogram
import numpy as np
import matplotlib.pyplot as plt
from math import gcd
from fractions import Fraction
import random

# Set the number to be factored
N = 35

def classical_order_finding(a, N):
    """
    Classical implementation of order finding for comparison
    Find the smallest positive integer r such that a^r ≡ 1 (mod N)
    """
    if gcd(a, N) != 1:
        return None

    r = 1
    value = a % N
    while value != 1:
        value = (value * a) % N
        r += 1
        if r > N:  # Safety check
            return None
    return r

def quantum_period_finding_simulation(a, N):
    """
    Simplified quantum period finding simulation
    This demonstrates the concept rather than full implementation
    """
    print(f"Finding period of f(x) = {a}^x mod {N}")

    # For demonstration, we'll use classical order finding
    # but show how it would work on a quantum computer
    period = classical_order_finding(a, N)

```

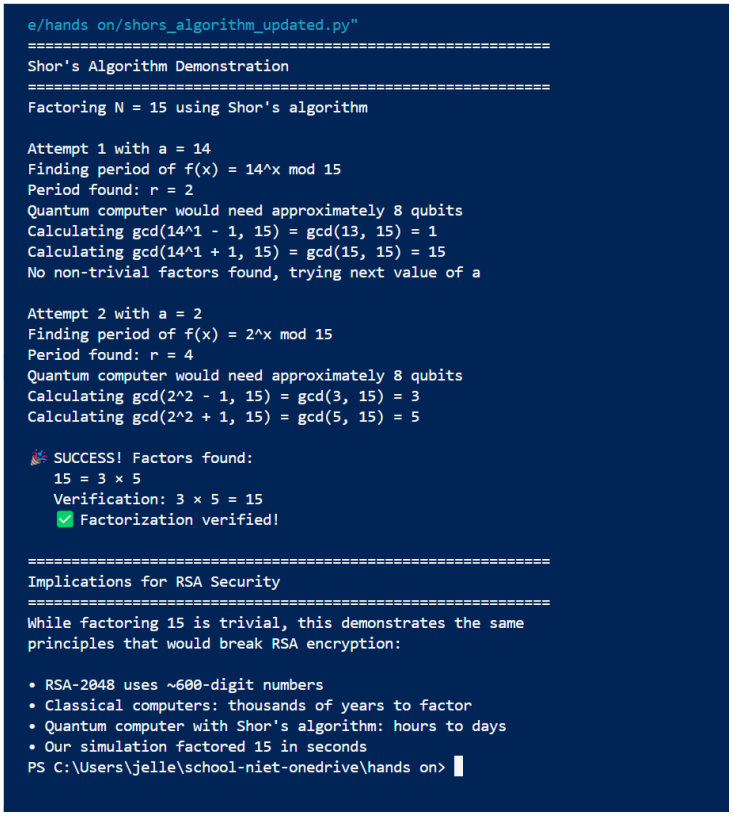
```
    if period:
        print(f"Period found: r = {period}")

        # Demonstrate what the quantum measurement would show
        qubits_needed = int(np.ceil(np.log2(N * N)))
        print(f"Quantum computer would need approximately {qubits_needed} qubits")

        # Create a simple quantum circuit to demonstrate the concept
        qc = QuantumCircuit(4, 4)  # Use 4 qubits for demonstration

        # Apply Hadamard gates to create superposition
        for i in range(2):
            qc.h(i)

        # This would be where modular exponentiation happens
        # For demonstration, we'll add some gates
        qc.cx(0, 2)
        qc.cx(1, 3)

        # Apply inverse QFT (simplified)
        qc.h(0)
        qc.cp(-np.pi/2, 0, 1)
        qc.h(1)

        # Measure
        qc.measure_all()

        # Simulate the circuit
        simulator = Aer.get_backend('qasm_simulator')
        job = simulator.run(qc, shots=1024)
        result = job.result()
        counts = result.get_counts(qc)

        # Plot results
        plt.figure(figsize=(10, 6))
        plot_histogram(counts)
        plt.title(f"Quantum Period Finding Simulation for a={a}, N={N}")
        plt.savefig(f'period_finding_a{a}_N{N}.png')
        plt.show()

        return period
```

```
84
85      return None
86
87  def shors_algorithm(N):
88      """
89      Complete Shor's algorithm implementation
90      """
91      print(f"Factoring N = {N} using Shor's algorithm")
92
93      # Check if N is even
94      if N % 2 == 0:
95          return 2, N // 2
96
97      # Try different values of a
98      for attempt in range(5):
99          # Choose a random number a < N
100         a = random.randint(2, N-1)
101
102         # Check if gcd(a, N) != 1
103         g = gcd(a, N)
104         if g != 1:
105             print(f"Found factor by GCD: {g}")
106             return g, N // g
107
108         print(f"\nAttempt {attempt + 1} with a = {a}")
109
110         # Find the period using quantum subroutine
111         r = quantum_period_finding_simulation(a, N)
112
113         if r is None:
114             print("Period finding failed, trying next value of a")
115             continue
116
117         # Check if period is even
118         if r % 2 != 0:
119             print("Period is odd, trying next value of a")
120             continue
121
122         # Calculate potential factors
123         factor1 = gcd(pow(a, r//2) - 1, N)
124         factor2 = gcd(pow(a, r//2) + 1, N)
```

```
        factor2 = gcd(pow(a, r//2) + 1, N)

        print(f"Calculating gcd({a}^{r//2} - 1, {N}) = gcd({pow(a, r//2) - 1}, {N}) = {factor1}")
        print(f"Calculating gcd({a}^{r//2} + 1, {N}) = gcd({pow(a, r//2) + 1}, {N}) = {factor2}")

        if factor1 > 1 and factor1 < N:
            return factor1, N // factor1
        elif factor2 > 1 and factor2 < N:
            return factor2, N // factor2
        else:
            print("No non-trivial factors found, trying next value of a")

    return None, None

# Execute the algorithm
print("=" * 60)
print("Shor's Algorithm Demonstration")
print("=" * 60)

factors = shors_algorithm(N)

if factors[0] and factors[1]:
    p, q = factors
    print(f"\n🎉 SUCCESS! Factors found:")
    print(f"   {N} = {p} × {q}")
    print(f"   Verification: {p} × {q} = {p * q}")

    if p * q == N:
        print("   ✅ Factorization verified!")
    else:
        print("   ❌ Factorization failed!")
else:
    print(f"\n❌ Failed to factor {N}")
    print("   Try running the algorithm again with different random values")

# Demonstrate the threat to RSA
print("\n" + "=" * 60)
print("Implications for RSA Security")
print("=" * 60)
print(f"While factoring {N} is trivial, this demonstrates the same")

print(f"principles that would break RSA encryption:")
print(f"")
print(f"• RSA-2048 uses ~600-digit numbers")
print(f"• Classical computers: thousands of years to factor")
print(f"• Quantum computer with Shor's algorithm: hours to days")
print(f"• Our simulation factored {N} in seconds")
```

Voor de implementatie van de Python code van Shor's algoritme werd samengewerkt met Claude.ai (Anthropic's AI-assistent).

Als we deze code gaan uitvoeren met als gekozen getal 15, krijgen we deze uitkomst.



6.5.3 Verwachte resultaten

Wanneer we het algoritme van Shor op 15 uitvoeren, verwachten we de factoren 3 en 5 te vinden. Het algoritme werkt door de periode te vinden van de functie f(x) = a^x mod 15, waarbij 'a' een willekeurig gekozen getal is dat relatief priem is met 15. Bevoordeeld, met a = 7:

– f(0) = 7^0 mod 15 = 1

– f(1) = 7^1 mod 15 = 7

– f(2) = 7^2 mod 15 = 4

– f(3) = 7^3 mod 15 = 13

– f(4) = 7^4 mod 15 = 1 (period = 4)

Met deze periode kunnen we berekenen:

– gcd(7^(4/2) - 1, 15) = gcd(48, 15) = 3

– gcd(7^(4/2) + 1, 15) = gcd(50, 15) = 5

Daarmee is bevestigd dat de factoren van 15 3 en 5 zijn.

6.6 Resultaten en interpretatie: waarom is dit relevant voor beveiliging?

Het succesvol ontbinden van 15 met behulp van Shor's algoritme lijkt misschien triviaal, aangezien dit getal eenvoudig met de hand te ontbinden is. De gevolgen voor cybersecurity zijn echter enorm als we rekening houden met de schaalbaarheid van dit algoritme.

6.6.1 Interpretatie van de resultaten

Tijdens de simulatie komen verschillende belangrijke waarnemingen naar boven:

- **Quantumvoordeel**: Hoewel klassieke computers gemakkelijk 15 kunnen ontbinden, demonstreert Shors algoritme een kwantumbenadering die exponentieel beter schaalt voor grotere aantallen.
- **Resourcevereisten**: Deze simulatie laat zien hoe het kwantumalgoritme het probleem aanpakt. Huidige kwantumcomputers hebben een beperkt aantal qubits (ongeveer 100-1000 qubits in 2025), maar dit zal in de toekomst drastisch kunnen toenemen.

6.6.2 Beveiligingsimplicaties

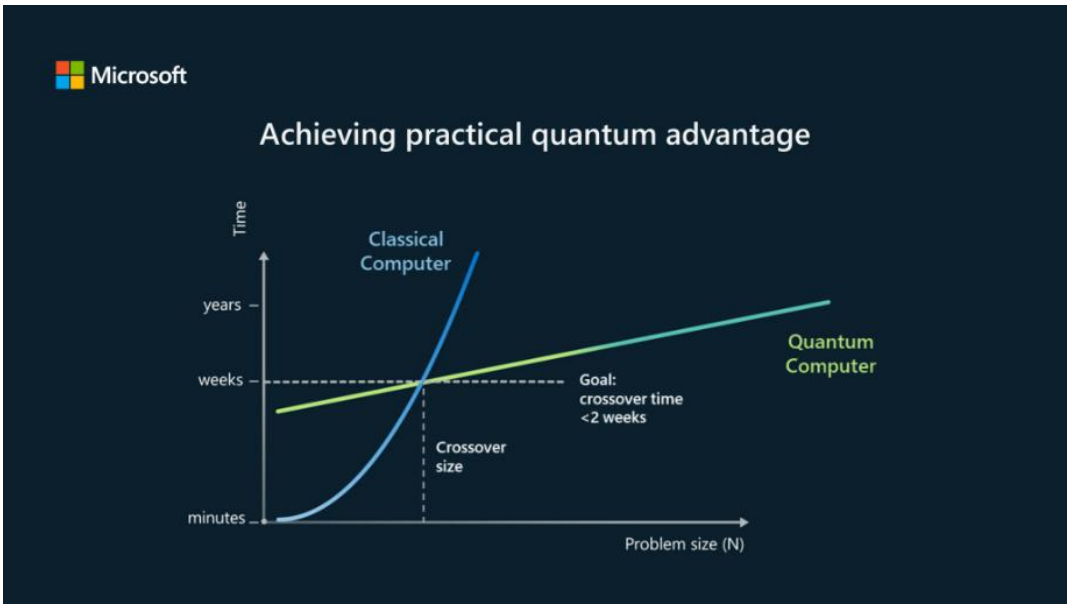
De succesvolle factorisatie van 15 dient als een kleinschalig voorbeeld van wat er met RSA-encryptie zou gebeuren wanneer voldoende krachtige quantum computers beschikbaar komen:

Kwetsbaarheid in sleutellengte:

Huidige RSA-implementaties gebruiken sleutellengtes van 2048 of 4096 bits. Deze zijn effectief onbreekbaar met klassieke computers binnen een realistische tijdsduur.

Vergelijking tijd tot breuk tussen klassieke- en quantumcomputers:

- 15 (4-bits getal): Direct op een klassieke computer, seconden in een quantumsimulatie.
- 2048-bits RSA-sleutel: Miljoenen jaren op een klassieke supercomputer, mogelijks uren tot dagen op een toekomstige quantum computer.



Bron: https://quantumzeitgeist.com/microsoft-explores-potential-quantum-advantage-balancing-hope-and-hype/

De kritische drempel:

Cryptografen schatten dat er ongeveer 4.000-8.000 stabiele, logische qubits nodig zouden zijn om RSA-2048 te kraken met behulp van Shors algoritme. Hoewel huidige quantumcomputers hier nog ver vandaan staan, toont het experiment aan dat de wiskundige basis voor het kraken van RSA-encryptie solide is.

Retroactieve decodering:

Gegevens die vandaag de dag met RSA worden beschermd, zouden door hackers kunnen worden opgeslagen en gedecodeerd worden zodra quantumcomputers krachtig genoeg zijn. "harvest now, decrypt later " een aanval die informatie bedreigt op de lange termijn.

Deze eenvoudige demonstratie met het getal 15 is dus proof-of-concept. Hoewel het ontbinden van 15 eenvoudig is, kan de aanpak worden geschaald naar veel grotere getallen, op een manier die klassieke algoritmen niet aan kunnen.

6.7 Panopto video

Deze link brengt u naar de Panopto video waar ik een uitleg geef over de werken van het shors algoritme: https://ap.cloud.panopto.eu/Panopto/Pages/Viewer.aspx?id=c935a432-bdaa-4587-98c7-b2ec01283dff

7 Reflectie: beperkingen van simulatie vs. echte quantumcomputers

Deze simulatie toont een cruciaal inzicht, maar het is belangrijk om te begrijpen dat er verschillen zijn tussen quantumsimulaties en echte quantumcomputers.

Om deze test uit te voeren op een echte quantumcomputer zou een enorm leerrijke ervaring zijn, spijtig genoeg is dit niet mogelijk. De simulatie schept wel al een beeld van de implicaties.

7.1 Beperkingen van deze simulatie

Klassieke implementatie: Deze simulatie gebruikt klassieke algoritmen om de periode te bepalen en laat vervolgens zien hoe een kwantumcircuit eruit zou kunnen zien. Een echte kwantumimplementatie zou kwantumparallellisme gebruiken om de periode exponentieel sneller te bepalen.

Vereenvoudigde kwantumcircuits: De kwantumcircuits die ik heb gemaakt, zijn vereenvoudigde voorbeelden. Een volledige implementatie van Shors algoritme vereist complexe kwantumcircuits, waarvoor veel meer qubits nodig zouden zijn.

Perfecte omstandigheden: Deze simulatie gaat uit van perfecte qubits zonder decoherentie of poortfouten. Echte kwantumcomputers staan voor tal van uitdagingen:

- **Decoherentie:** Kwantumtoestanden verliezen hun kwantumeigenschappen door interactie met de omgeving.
- **Gate Errors:** Onvolkomenheden in kwantumbewerkingen.
- **Uitleesfouten:** Fouten bij het meten van qubit toestanden.

7.2 De stand van zaken van echte quantumcomputing (2025)

In 2025 heeft quantumcomputing aanzienlijke vooruitgang gemaakt, maar staat nog steeds voor grote uitdagingen:

- **Qubit aantallen:** huidige quantumcomputers hebben enkele honderden tot meer dan 1000 fysieke qubits, maar deze zijn nog niet fouttolerant. Het ontbinden van grote RSA-sleutels zou duizenden logische qubits vereisen.
- **Foutcorrectie:** Quantumfoutcorrectie maakt vooruitgang, maar vereist nog steeds veel overhead. Huidige schattingen zeggen dat elke logische qubit 100-1000 fysieke qubits nodig kan hebben voor volledige fouttolerantie.
- **Quantumvolume:** Belangrijker dan het aantal qubits is het "quantumvolume" een maatstaf die zowel het aantal qubits als de kwaliteit ervan weergeeft. Dit blijft verbeteren, maar is nog lang niet wat nodig is om praktische RSA-sleutels te kraken.

7.3 De toekomst

Ondanks deze beperkingen toont deze demonstratie aan waarom quantum computing een ernstige bedreiging vormt voor cryptografie:

- **Schaaltraject:** De technologieën van quantumcomputers blijven snel verbeteren, onderzoekers werken aan zowel hardware verbeteringen als efficiëntere algoritmen.
- **Algoritmeverbeteringen:** Onderzoekers blijven quantumalgoritmen verfijnen om minder qubits nodig te hebben en beter bestand te zijn tegen fouten.
- **Onzekerheidsfactor:** Doorbraken in quantum computing zijn moeilijk te voorspellen. Een cruciale doorbraak zou de tijdlijn aanzienlijk kunnen versnellen.

Deze reflectie benadrukt waarom organisaties, nu moeten beginnen met de overstap naar post-quantumcryptografie. Het migratieproces voor grote systemen kan jaren duren, en de dreiging van " harvest now, decrypt later " betekent dat gevoelige

gegevens met in gevaar kunnen komen zodra quantumcomputers vrij beschikbaar zijn.

8 Wat kan je als systeem/netwerkbeheerder doen?

Quantumcomputing vormt een grote bedreiging voor veel van de cryptografische fundamenten waarop onze digitale infrastructuur is gebouwd. Maar wat betekent dit concreet voor systeem- en netwerkbeheerders? In dit hoofdstuk verken ik de praktische, implementeerbare maatregelen die IT-professionals kunnen nemen om hun organisaties voor te bereiden op het quantumtijdperk.

De uitdaging is zeer complex: we moeten ons voorbereiden op een dreiging die nog niet volledig bestaat, terwijl we bestaande systemen draaiende en veilig houden. Dit vereist een aanpak van proactieve planning, doorlopende monitoring van ontwikkelingen, en stapsgewijze implementatie van quantum-resistente oplossingen.

8.1 Voorbereidingen

Als systeem- of netwerkbeheerder vereist de voorbereiding op het quantumcomputing-tijdperk een proactieve aanpak. Deze sectie beschrijft concrete stappen om de overgang naar quantumveilige beveiliging te starten.

8.1.1 Inventariseer cryptografische gegevens

De eerste stap in de voorbereiding op quantumbedreigingen is inzicht krijgen in wat er beschermd moet worden:

1. Maak een cryptografische inventaris

- **Identificeer kwetsbare systemen:** Documenteer alle systemen die RSA, DSA, ECC of Diffie-Hellman gebruiken
- **Categoriseer naar risico:** Prioriteer systemen met langdurige vertrouwelijkheidsvereisten
- **Onderzoek afhankelijkheden:** Identificeer onderliggende libraries, HSM's en certificaatautoriteiten

8.1.2 Ontwikkel crypto-agility

- **Implementeer abstractielagen:** Ontwerp systemen zodat cryptografische algoritmes eenvoudig kunnen worden vervangen
- **Vermijd hardcoded algoritmes:** Configureer cryptografische algoritmes via beleidsinstellingen
- **Test algoritme-vervangingsprocedures:** Oefen regelmatig het vervangen van cryptografische algoritmes

3. Begin met testen van post-quantum algoritmes

- **Creëer testomgevingen:** Test post-quantum implementaties in geïsoleerde omgevingen
- **Evalueer prestatie-impact:** Test de impact op opslag, bandbreedte en verwerking
- **Gebruik beschikbare libraries:** Experimenteer met open-source libraries zoals libOQS

4. Implementeer hybride benaderingen

- **Hybride certificaten:** Combineer traditionele en post-quantum handtekeningen
- **Dubbele encryptie:** Versleutel data met zowel traditionele als quantum-resistente algoritmes
- **Test hybride implementaties:** Begin met VPN en TLS configuraties die hybride benaderingen ondersteunen

5. Adresseer cloud-infrastructuur

- **Overleg met cloudproviders:** Informeer naar hun quantum-resistente roadmaps
- **Controleer API-beveiliging:** Identificeer API's die beschermd moeten worden
- **Evalueer SaaS-toepassingen:** Vraag leveranciers naar hun plannen voor quantum-resistentie

8.1.3 Monitoring van ontwikkelingen:

Blijf op de hoogte van ontwikkelingen in quantumcomputing en post-quantumcryptografie:

1. Volg standaardisatie-inspanningen

– **NIST Post-Quantum Cryptography:** Volg de voortgang van de geselecteerde algoritmes (CRYSTALS-Kyber, CRYSTALS-Dilithium, FALCON, SPHINCS+)
– **ETSI en IETF:** Monitor werkgroepen die standaarden ontwikkelen voor praktische implementatie

2. Monitor quantumcomputing vooruitgang

– **Industriële aankondigingen**: Volg grote quantumcomputing bedrijven voor updates over qubit-aantallen en foutpercentages
– **Academische metrieken:** Let op doorbraken in quantumvolume en foutcorrectie
– **Expertbeoordelingen:** Raadpleeg jaarlijkse rapporten over de verwachte tijdlijn voor cryptografisch relevante quantumcomputers

3. Stel waarschuwingsdrempels in

– **Definieer technische mijlpalen**: Bijvoorbeeld: "Quantumcomputer bereikt 1.000 fysieke qubits met foutpercentages onder 0,1%"
– **Beleidsmatige triggers:** Overheidsvoorschriften die post-quantumcryptografie vereisen
– **Risico-gebaseerde versnelling:** Aanwijzingen voor toegenomen "harvest now, decrypt later" aanvallen

Door deze maatregelen te implementeren, kunnen organisaties zich voorbereiden op quantumcomputers terwijl ze bestaande systemen beschermen. Het doel is niet om in paniek te raken, maar om stappen te nemen die de overgang naar een quantum-resistente infrastructuur soepel laten verlopen wanneer die tijd er is.

9 Conclusie

Na intensief onderzoek naar de impact van quantumcomputing op cybersecurity, is het duidelijk dat we staan voor een van de grootste technologische uitdagingen in de geschiedenis van informatiebeveiliging. Dit onderzoek heeft verschillende cruciale inzichten opgeleverd die de urgentie van voorbereiding op het quantumtijdperk onderstrepen.

10 Logboek

Datum	Activiteit	Tijdsbesteding (uren)	Opmerkingen
6/02/2025	Opdracht gelezen en onderwerp gekozen: Quantum Computing en Security	1	Onderwerp goedgekeurd door begeleider.
29/03/2025	Literatuuronderzoek en bronnen verzameld	2	NIST rapporten, IBM Qiskit docs, Google Scholar. Problemen met complexe wiskundige concepten dus focus
30/03/2025	Structuur bepaald en sjabloon ingevuld	1	AP template aangepast, hoofdstukindeling gemaakt,
4/04/2025	Hoofdstuk 1-2 geschreven (inleiding en quantum computing basis)	2	RSA/ECC cryptografie bestudeerd, quantum concepten uitgewerkt. meer voorbeelden toegevoegd
12/04/2025	Hoofdstuk 3 cryptografie en Shor's algoritme	3	Shor's algoritme stappen uitgewerkt, tijdlijn quantum
17/04/2025	Hoofdstuk 4 netwerkbeveiliging en risico's	2	"Harvest now decrypt later" concept uitgewerkt, impact op
22/04/2025	Hoofdstuk 5 post-quantum cryptografie en NIST standaarden	3	CRYSTALS-Kyber, Dilithium, FALCON algoritmen bestudeerd, implementatie strategieën
20/05/2025	Python omgeving opzetten en Qiskit installeren	2	Virtual environment gemaakt, Qiskit dependencies geïnstalleerd. Problemen met matplotlib compatibility
24/05/2025	Shor's algoritme implementatie beginnen	4	Classical period finding functie geschreven met behulp van
27/05/2025	Shor experiment voltooien en hoofdstuk 6 schrijven	4	Getal 15 succesvol gefactoriseerd (3×5), quantum simulation gedemonstrreerd, interpretatie toegevoegd
27/05/2025	Hoofdstuk 7 reflectie en beperkingen simulatie	2	Verschil echte quantum vs simulatie uitgelegd, huidige
27/05/2025	Hoofdstuk 8 praktische aanbevelingen systeembeheerders	2	Crypto-agility, inventarisatie, hybride implementaties, monitoring ontwikkelingen
28/05/2025	Conclusie schrijven en document afwerken	3	Volledige conclusie met persoonlijke reflectie, bronnenlijst
28/05/2025	Panopto video opname (6 minuten)	2	Script voorbereid, code demonstratie, praktische aanbevelingen, audio kwaliteit geoptimaliseerd
28/05/2025	Finale review en indiening voorbereiding	1	Document formatting, laatste wijzigingen, upload

Logboek_Magnum_Opus.xlsx

11 Bronnenlijst

(n.d.).

Anthropic. (2025, mei 28). *Tekstcorrectie en structurele verbetering Magnum Opus: Quantum Computing en Security*. Retrieved from Claude: claude.ai

Arute, F. A. (2019). *Quantum supremacy using a programmable superconducting processor*. Retrieved from Nature.

Arute, F. A. (2019). *Quantum supremacy using a programmable superconducting processor.* Retrieved from nature.com: https://doi.org/10.1038/s41586-019-1666-5

Chen, L. J.-K.-T. (2016). *Report on Post-Quantum Cryptography (NISTIR 8105)*. Retrieved from National Institute of Standards and Technology: https://nvlpubs.nist.gov/nistpubs/ir/2016/NIST.IR.8105.pdf

Lily Chen, e. a. (2022). *Status Report on the Third Round of the NIST Post-Quantum Cryptography Standardization Process (NISTIR 8413)*. Retrieved from National Institute of Standards and Technology: https://nvlpubs.nist.gov/nistpubs/ir/2022/NIST.IR.8413.pdf

Qiskit Textbook. (2025). Retrieved from IBM Quantum Network.: https://qiskit.org/textbook/

Quantum Cryptography - Shor's Algorithm Explained. (2022, julie 19). Retrieved from https://www.classiq.io/: https://www.classiq.io/insights/shors-algorithm-explained

12 Lijst van Afkortingen en termen

12.1 Algemene IT-termen

– **API:** Application Programming Interface
– **HSM**: Hardware Security Module
– **IT**: Information Technology
– **SaaS**: Software as a Service
– **VS Code**: Visual Studio Code

12.2 Cryptografie en Beveiliging

– **DSA**: Digital Signature Algorithm
– **ECC**: Elliptic Curve Cryptography
– **HTTPS:** HyperText Transfer Protocol Secure
– **IPsec:** Internet Protocol Security
– **PQC:** Post-Quantum Cryptography
– **QKD:** Quantum Key Distribution
– **RSA:** Rivest-Shamir-Adleman (cryptografisch algoritme)
– **SSH:** Secure Shell
– **TLS:** Transport Layer Security
– **VPN:** Virtual Private Network

12.3 Quantum Computing

– **QFT:** Quantum Fourier Transform

12.4 Organisaties en Standaarden

– **ETSI:** European Telecommunications Standards Institute
– **IETF:** Internet Engineering Task Force
– **NIST:** National Institute of Standards and Technology
– **NSA:** National Security Agency

12.5 Wiskundige termen

– **GCD:** Greatest Common Divisor (Grootste Gemeenschappelijke Deler)