Building and Breaking Lattice-Based Post-Quantum Cryptosystem Hardware

Aydin Aysu, Assistant Professor

Researchers: Dr. Seetal Potluri, Anuj Dubey, Furkan Aydin, Emre Karabulut, Priyank Kashyap, Gregor Haas, Faiz Alam, Ferhat Yaman

aaysu@ncsu.edu

Research Sponsors
Our Research

Cybersecurity with a *hardware* focus

- Hardware acceleration for next-generation cryptography
  [DATE’20][TC’20][FPL’20][ICCAD’20][ESL’19][TC’15][TECS’15][ESL’14] [HOST’13]
- Hardware building blocks to combat supply-chain attacks
  [ICCAD’21] [HOST’18][HOST’17][ICISC’16][DATE’16][CHES’15][WESS’13]
- Mitigating hardware theft of untrusted foundries
  [TCAD’22][ISQED’20][ISCAS’20][TCAD’22]
- Implementation security: side-channel and fault attacks
  [DAC’22][HOST’22][TCHES’22][DAC’21][HOST’20][ICCAD’20][HOST’18][DATE’14]
- Training a cyber-aware STEM workforce
  [GLS-VLSI’22][GLS-VLSI’19]
Why Quantum Computing?

- Better predicting tomorrow’s weather?
- Efficient simulation of chemical reactions?
- Finding new electronic materials?
- Optimize traffic, logic simulations, or ticket prices?
- …
Know This Machine?
Encryption for the web rely on hard mathematical problems

 Encryption of data relies on the difficulty of solving certain mathematical problems. For example, RSA-4096, a widely used encryption algorithm, requires a 4096-bit number to be factored. Classical computers can solve this problem at a significant cost of time:

- 100 bits: 1 second
- 1000 bits: 1 hour
- 10000 bits: 1 year
- 100000 bits: 1 million years
- 1 billion bits: 1 billion years

This table shows the state of the art and the time it would take for a classical computer to factor an n-bit number.
Quantum Computers Endanger Cryptography

Encryption for the web rely on hard mathematical problems

Time to Factor an n-bit Number

- 1 Sec
- 1 Hour
- 1 Year
- 1 Million Years
- 1 Billion Years

Classical Computer State of the Art

Classical Computer 30 Years from Now (Moore’s Law)

100 1000 10000 100000

n (Bits)

RSA-4096

[IBM-CAI’16]
Quantum Computers Endanger Cryptography

Encryption for the web rely on hard mathematical problems

Time to Factor an n-bit Number

Classical Computer
State of the Art

Classical Computer
30 Years from Now (Moore’s Law)

Quantum Computer

$O(n^3 \log n)$ with $2n+3$ qbits
Quantum Computers Endanger Cryptography

Encryption for the web rely on hard mathematical problems

Time to Factor an n-bit Number

- 1 Sec
- 1 Hour
- 1 Year
- 1 Million Years
- 1 Billion Years

Classical Computer State of the Art

Classical Computer 30 Years from Now (Moore’s Law)

Quantum Computer

\[ O(n^3 \log n) \] with 2n+3 qbits

Current State

- RSA-4096
- 100
- 1000
- 10000
- 100000
- 1 Million
- 1 Billion

[IBM-CAI’16]
Emergence of Post-Quantum Cryptography

- NIST’s PQ standardization effort (2017–2024)
- Some industry/government adoptions already occurred
Emergence of Post-Quantum Cryptography

• Key Encapsulation Mechanisms
  – CRYSTALS-KYBER

• Digital Signatures
  – CRYSTALS-DILITHIUM
  – FALCON
  – SPHINCS+

• Alternates:
  – FrodoKEM, NTRU, NTRU Prime, SABER, …
Migration to Post-Quantum Cryptography

The advent of quantum computing technology will compromise many of the current cryptographic algorithms, especially public-key cryptography, which is widely used to protect digital information. Most algorithms on which we depend are used worldwide in components of many different communications, processing, and storage systems. Once access to practical quantum computers becomes available, all public-key algorithms and associated protocols will be vulnerable to criminals, competitors, and other adversaries. It is critical to begin planning for the replacement of hardware, software, and services that use public-key algorithms now so that information is protected from future attacks.

Source: https://www.nccoe.nist.gov/
## Category of Post-Quantum Cryptosystems

<table>
<thead>
<tr>
<th>Category</th>
<th>Security Assumption</th>
<th>Features</th>
</tr>
</thead>
<tbody>
<tr>
<td>Code-based cryptography</td>
<td>Decoding general linear codes</td>
<td>Large keys, complex operations</td>
</tr>
<tr>
<td>Hash-based cryptography</td>
<td>One-way hash functions</td>
<td>Large keys, limited applications</td>
</tr>
<tr>
<td>Lattice-based cryptography</td>
<td>Lattice problems</td>
<td>Small keys, efficient arithmetic,</td>
</tr>
</tbody>
</table>

3 out of the 4 upcoming NIST standards use lattice cryptography
Lattices Have Other Uses...

Homomorphic encryption allows computing on encrypted data without knowing the secret key or underlying plaintext.

\[ Enc(pk, m) \]

\[ r = Dec(sk, c) \]
Security of Lattice-Based Cryptography

Given a **bad basis**, can you find a **good one**?
Lattice-based Cryptography

• A Lattice is a set of points
  \[ L = \{ a_1 v_1 + \ldots + a_n v_n \mid a_i \text{ integers} \} \]
  with \( v_1, \ldots, v_n \) in \( \mathbb{R}^n \) linearly independent

• Approximate Shortest Vector Problem (SVP):
  Given basis \( v_0, v_1 \) find a short vector \( \lambda_1 \)

• NP-Hard [Ajtai’96]

• Lattice basis reduction attack complexities
  – Classical: \( 2^{2n+o(n)} \) [MV’10]
  – Quantum: \( 2^{1.799n+o(n)} \) [LMP’13]
Trap-door one-way function

Learning With Errors:
\[ B = A \cdot S + E, \text{ PUBLIC KEY} = B \text{ and } A, \text{ SECRET KEY} = S \]

Works with matrices and polynomials
Fundamental Computations in Lattice-Based Cryptography

Matrix Multiplication

\[
\begin{bmatrix}
2 & 13 & 7 & 3 \\
4 & 7 & 9 & 1 \\
6 & 14 & 5 & 11
\end{bmatrix}
\cdot
\begin{bmatrix}
8 & 1 \\
3 & -1 \\
12 & 2 \\
5
\end{bmatrix}
= 
\begin{bmatrix}
13 \ \\
12 \ \\
3
\end{bmatrix}
\]

Polynomial Multiplication

\[
\begin{bmatrix}
2 & 13 & 3 \\
13 & 7 & 12 \\
3 & 5
\end{bmatrix}
\cdot
\begin{bmatrix}
8 & 1 \\
3 & -1 \\
5 & -1
\end{bmatrix}
= 
\begin{bmatrix}
8 & 1 \\
16 & 6
\end{bmatrix}
\]

Elements are defined over Galois Field (modular arithmetic with primes)
Random sampling may require “discrete Gaussian” distributions
FALCON Specification – What to Implement?

Algorithm 5 \texttt{NTRUGen}(\phi, q)

Require: A monic polynomial \( \phi \in \mathbb{Z}[x] \) of degree \( n \), a modulus \( q \)
Ensure: Polynomials \( f, g, F, G \)

1: \( \sigma_{\{f,g\}} \leftarrow 1.17\sqrt{q/2n} \)
2: for \( i \) from 0 to \( n - 1 \) do
3: \[
\begin{align*}
    f_i & \leftarrow D_{\mathbb{Z},\sigma_{\{f,g\}},0} \\
    g_i & \leftarrow D_{\mathbb{Z},\sigma_{\{f,g\}},0}
\end{align*}
\]
4: \( f \leftarrow \sum_i f_i x^i \)
5: \( g \leftarrow \sum_i g_i x^i \)
6: if \( \text{NTT}(f) \) contains 0 as a coefficient then
   restart
7: \( \gamma \leftarrow \max \left\{ \|\gamma^*\|, \left\| \left( \frac{qf^*}{ff^*+gg^*}, \frac{qg^*}{ff^*+gg^*} \right) \right\| \right\} \)
8: if \( \gamma > 1.17\sqrt{q} \) then
   restart
9: \( F, G \leftarrow \text{NTRUSolve}_{n,q}(f, g) \)
10: if \( (F, G) = \perp \) then
    restart
11: return \( f, g, F, G \)

\( \triangleright \) \( \sigma_{\{f,g\}} \) is chosen so that \( \mathbb{E}[\| (f, g) \|] = 1.17\sqrt{q} \)
\( \triangleright \) See also (3.29)
\( \triangleright \) \( f \in \mathbb{Z}[x]/(\phi) \)
\( \triangleright \) \( g \in \mathbb{Z}[x]/(\phi) \)
\( \triangleright \) Check that \( f \) is invertible mod \( q \)
\( \triangleright \) Using (3.9) with (3.8) or (3.10)
\( \triangleright \) Check that \( \gamma = \| B \|_{GS} \) is short
\( \triangleright \) Computing \( F, G \) such that \( fG - gF = q \mod \phi \)
New IPs Needed for Lattice-Based Cryptography

- Building blocks for discrete Gaussian sampling
- Building blocks for Number Theoretic Transform
- Full system design working with new building blocks
- System-level trade-offs
- Optimizations for edge computers to cloud
- New custom instructions for ISA
- Implementation security!
- Hybrid designs
Number Theoretic Transform

Schoolbook Method

NTT-based Method

Reduces multiplication complexity from $O(n^2)$ to $O(n \cdot \log n)$
Number Theoretic Transform

Iterative NTT Algorithm

Algorithm 2 Iterative NTT Algorithm [14]

Input:  \( A(x) \in \mathbb{Z}_q[x]/(x^n + 1) \)

Input:  primitive \( n \)-th root of unity \( \omega \in \mathbb{Z}_q \), \( n = 2^l \)

Output:  \( \overline{A}(x) = \text{NTT}(A) \in \mathbb{Z}_q[x]/(x^n + 1) \)

1:  for \( i \) from 1 by 1 to \( l \) do
2:    \( m = 2^{l-i} \)
3:    for \( j \) from 0 by 1 to \( 2^{i-1} - 1 \) do
4:      for \( k \) from 0 by 1 to \( m - 1 \) do
5:        \( U \leftarrow A[2 \cdot j \cdot m + k] \)
6:        \( V \leftarrow A[2 \cdot j \cdot m + k + m] \)
7:        \( A[2 \cdot j \cdot m + k] \leftarrow U + V \)
8:        \( A[2 \cdot j \cdot m + k + m] \leftarrow \omega^{(2^{i-1} \cdot k)} \cdot (U - V) \)
9:      end for
10:    end for
11:  end for
12: return \( A \)

- N-point NTT operation has \( \log_2 n \) stages
- At each stage, \( n/2 \) butterfly operation is performed
- Single NTT operation can be parallelized using multiple butterfly units
Number Theoretic Transform

Algorithm 3 Word-Level Montgomery Reduction Algorithm for NTT-friendly primes

Input: $C = A \cdot B$ (a 2K-bit positive integer)
Input: $q$ (a K-bit modulus), $q = q_H \cdot 2^w + 1$
Input: $w = \log_2(2n)$ (word size)
Output: $Res = C \cdot R^{-1} \pmod{q}$ where $R = 2^{w \cdot L} \pmod{q}$

1: $L = \lceil \frac{w}{2} \rceil$
2: $T1 = C$
3: for $i$ from 0 to $L$
4: $T1_H = T1 >> w$
5: $T1_L = T1 \pmod{2^w}$
6: $T2 = \text{two's complement of } T1_L$
7: $cin = T2[\lceil w - 1 \rceil \lor T1_L[\lceil w - 1 \rceil]$
8: $T1 = T1_H + (q_H \cdot T2[\lceil w - 1 \rceil : 0]) + cin$
9: end for
10: $T4 = T1 - q$
11: if $(T4 < 0)$ then $Res = T4$ else $Res = T4$

Number Theoretic Transform Results

<table>
<thead>
<tr>
<th>Met.</th>
<th>Work</th>
<th>Platform</th>
<th>$n$</th>
<th>$K$</th>
<th>LUT / REG / DSP / BRAM</th>
<th>Clock (MHz)</th>
<th>Latency</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>CC</td>
</tr>
<tr>
<td></td>
<td></td>
<td>512</td>
<td></td>
<td>240 / - / 3 / 2</td>
<td>-</td>
<td>-</td>
<td>50</td>
</tr>
<tr>
<td></td>
<td></td>
<td>1024</td>
<td></td>
<td>250 / - / 3 / 2</td>
<td>-</td>
<td>-</td>
<td>100</td>
</tr>
<tr>
<td>[21]$^a,b$</td>
<td>Virtex-6</td>
<td>256</td>
<td>13</td>
<td>4549 / 3624 / 1 / 12</td>
<td>262</td>
<td>-</td>
<td>8</td>
</tr>
<tr>
<td></td>
<td></td>
<td>4096</td>
<td>30</td>
<td>64K / - / 200 / 400</td>
<td>225</td>
<td>-</td>
<td>73</td>
</tr>
<tr>
<td>[23]$^b$</td>
<td>Virtex-7</td>
<td>1024</td>
<td>32</td>
<td>1208 / - / 14 / 14</td>
<td>212</td>
<td>-</td>
<td>12</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>34K / 16K / 476 / 228</td>
<td>200</td>
<td>80</td>
<td>0.4</td>
</tr>
<tr>
<td>[24]$^b$</td>
<td>Spartan-6</td>
<td>1024</td>
<td>32</td>
<td>67K / - / 599 / 129</td>
<td>200</td>
<td>140</td>
<td>0.7</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>77K / - / 952 / 325.5</td>
<td></td>
<td>80</td>
<td>0.4</td>
</tr>
<tr>
<td>[18]$^b$</td>
<td>Virtex-7</td>
<td>1024</td>
<td>32</td>
<td>1349 / 860 / 1 / 2</td>
<td>313</td>
<td>1691</td>
<td>5.4</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1536 / 953 / 1 / 3</td>
<td>278</td>
<td>3443</td>
<td>12.3</td>
</tr>
<tr>
<td></td>
<td></td>
<td>512</td>
<td>14</td>
<td>13</td>
<td>313</td>
<td>278</td>
<td>12.3</td>
</tr>
<tr>
<td></td>
<td></td>
<td>512</td>
<td>14</td>
<td>- / - / - / -</td>
<td>313</td>
<td>278</td>
<td>12.3</td>
</tr>
<tr>
<td>[26]$^c$</td>
<td>40nm CMOS</td>
<td>256</td>
<td>13</td>
<td>- / - / - / -</td>
<td>300</td>
<td>160</td>
<td>0.5</td>
</tr>
<tr>
<td></td>
<td></td>
<td>512</td>
<td>14</td>
<td>- / - / - / -</td>
<td>492</td>
<td>12.3</td>
<td></td>
</tr>
<tr>
<td>[27]$^c$</td>
<td>UMC 65nm</td>
<td>256</td>
<td>13</td>
<td>- / - / - / -</td>
<td>25</td>
<td>2056</td>
<td>82</td>
</tr>
<tr>
<td></td>
<td></td>
<td>512</td>
<td>14</td>
<td>- / - / - / -</td>
<td>4616</td>
<td>184</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>1024</td>
<td>14</td>
<td>- / - / - / -</td>
<td>10248</td>
<td>409</td>
<td></td>
</tr>
<tr>
<td>[28]$^a,b$</td>
<td>Artix-7</td>
<td>1024</td>
<td>14</td>
<td>4823 / 2901 / 8 / -</td>
<td>153</td>
<td>1280</td>
<td>-</td>
</tr>
<tr>
<td>[29]$^b$</td>
<td>Virtex-7</td>
<td>16384</td>
<td>32</td>
<td>2.81K / 1.25K / 39 / 80</td>
<td>168</td>
<td>28672</td>
<td>-</td>
</tr>
<tr>
<td></td>
<td></td>
<td>32768</td>
<td>32</td>
<td>2.86K / 1.27K / 39 / 160</td>
<td>166</td>
<td>61440</td>
<td>-</td>
</tr>
<tr>
<td>[30]$^b$</td>
<td>Virtex-6</td>
<td>65536</td>
<td>30</td>
<td>72K / 63K / 250 / 84</td>
<td>100</td>
<td>47795</td>
<td>-</td>
</tr>
<tr>
<td>TW-1 PE</td>
<td>Virtex-7</td>
<td>1024</td>
<td>14</td>
<td>575 / - / 11 / 11</td>
<td>125</td>
<td>5160</td>
<td>41.2</td>
</tr>
<tr>
<td>TW-8 PE</td>
<td>Virtex-7</td>
<td>1024</td>
<td>14</td>
<td>2720 / - / 31 / 180</td>
<td>125</td>
<td>24708</td>
<td>197.6</td>
</tr>
<tr>
<td>TW-32 PE</td>
<td>Virtex-7</td>
<td>1024</td>
<td>14</td>
<td>17188 / - / 96 / 48</td>
<td>125</td>
<td>200</td>
<td>1.6</td>
</tr>
</tbody>
</table>

High Precision Discrete Gaussian Sampling

\[ f(x) = \frac{1}{\sigma \sqrt{2\pi}} e^{-\frac{1}{2} \left( \frac{x-\mu}{\sigma} \right)^2} \]

Sampling precision impacts cryptographic security level

\[ p=0.241970724519143349\ldots \]

\[ 0.241970724519143348 \]

\[ 2^{128} \rightarrow 2^{56} \]
# High Precision Discrete Gaussian Sampling

<table>
<thead>
<tr>
<th>Sampler</th>
<th>Speed</th>
<th>FP exp()</th>
<th>Table Size</th>
<th>Table Lookup</th>
<th>Entropy</th>
<th>Features</th>
</tr>
</thead>
<tbody>
<tr>
<td>Rejection</td>
<td>slow</td>
<td>10</td>
<td>0</td>
<td>0</td>
<td>45+10log₂σ</td>
<td>Suitable for constrained devices</td>
</tr>
<tr>
<td>Ziggurat</td>
<td>flexible</td>
<td>flexible</td>
<td>flexible</td>
<td>flexible</td>
<td>flexible</td>
<td>Suitable for encryption requires high-precision FP arithmetic; not suitable for HW implementation</td>
</tr>
<tr>
<td>CDT</td>
<td>fast</td>
<td>0</td>
<td>στλ</td>
<td>log₂(τσ)</td>
<td>2.1+log₂σ</td>
<td>Suitable for digital signature easy to implement</td>
</tr>
<tr>
<td>Knuth-Yao</td>
<td>fastest</td>
<td>0</td>
<td>1/2στλ</td>
<td>log₂(√2πeσ)</td>
<td>2.1+log₂σ</td>
<td>Not suitable for digital signature</td>
</tr>
<tr>
<td>Bernoulli</td>
<td>fast</td>
<td>0</td>
<td>λlog₂(2.4τσ²)</td>
<td>≈ log₂σ</td>
<td>≈ 6 + 3log₂σ</td>
<td>Suitable for all schemes</td>
</tr>
<tr>
<td>Binomial</td>
<td>fast</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>4σ²</td>
<td>Not suitable for digital signature</td>
</tr>
</tbody>
</table>

Many algorithmic options for implementing Gaussian sampling
High Precision Discrete Gaussian Sampling

(a) The Proposed Gaussian Sampler Hardware’s Top Level Block Diagram

(b) Fusion Tree Search’s Block Diagram

## Results and Comparison

<table>
<thead>
<tr>
<th>Work</th>
<th>Supported Algorithms</th>
<th>$\sigma/\lambda/\text{Depth}$</th>
<th>Platform</th>
<th>Slice/FFs/BRAM</th>
<th>$F_{\text{Max}}$ (MHz)</th>
<th>Cyc Cnt</th>
<th>Area-Delay</th>
</tr>
</thead>
<tbody>
<tr>
<td>HW [15]</td>
<td>qTESLA p-I</td>
<td>8.5/64/77</td>
<td>Artix-7</td>
<td>907/812/3</td>
<td>115</td>
<td>111</td>
<td>235.88×</td>
</tr>
<tr>
<td>This Work</td>
<td></td>
<td></td>
<td>Virtex-7</td>
<td>169/554/306/60</td>
<td>232</td>
<td>-</td>
<td></td>
</tr>
<tr>
<td>HW [15]</td>
<td>qTESLA p-III</td>
<td>8.5/125/110</td>
<td>Artix-7</td>
<td>820/837/3</td>
<td>119</td>
<td>49</td>
<td>35.26×</td>
</tr>
<tr>
<td>This Work</td>
<td></td>
<td></td>
<td>Virtex-7</td>
<td>324/1049/566/0</td>
<td>162</td>
<td>-</td>
<td></td>
</tr>
<tr>
<td>HW [11]</td>
<td>LP</td>
<td>3.33/64/31</td>
<td>Virtex-6</td>
<td>43/112/19/0</td>
<td>297</td>
<td>5</td>
<td>0.16×</td>
</tr>
<tr>
<td>HW [10]</td>
<td></td>
<td>3.33/90/37</td>
<td>Virtex-5</td>
<td>17/43/33/1</td>
<td>259</td>
<td>3</td>
<td>0.36×</td>
</tr>
<tr>
<td>HW [8]</td>
<td></td>
<td>3.33/80/35</td>
<td>Virtex-6</td>
<td>231/863/6/0</td>
<td>61</td>
<td>1</td>
<td>1.06×</td>
</tr>
<tr>
<td>This Work</td>
<td>LP</td>
<td>3.33/64/33</td>
<td>Virtex-7</td>
<td>360/1278/306/0</td>
<td>218</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td></td>
<td></td>
<td>3.33/90/37</td>
<td></td>
<td>442/1418/306/0</td>
<td>198</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td></td>
<td></td>
<td>3.33/80/35</td>
<td></td>
<td>425/1341/8/0</td>
<td>205</td>
<td>2</td>
<td>-</td>
</tr>
<tr>
<td></td>
<td></td>
<td>3.33/100/39</td>
<td></td>
<td>539/1960/446/0</td>
<td>173</td>
<td>2</td>
<td>-</td>
</tr>
<tr>
<td>HW$^a$ [11]</td>
<td>BLISS</td>
<td>215/64/184</td>
<td>Spartan-6</td>
<td>179/577/64/0</td>
<td>130</td>
<td>8</td>
<td>1.67×</td>
</tr>
<tr>
<td>This Work</td>
<td></td>
<td></td>
<td>Spartan-6</td>
<td>299/928/1121/0</td>
<td>129</td>
<td>8</td>
<td>2.85×</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Virtex-7</td>
<td>305/1001/558/0</td>
<td>245</td>
<td>5</td>
<td>-</td>
</tr>
<tr>
<td>This Work</td>
<td>FrodoKEM-640</td>
<td>2.8/16/12</td>
<td>Virtex-7</td>
<td>71/203/106/0</td>
<td>292</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>FrodoKEM-976</td>
<td>2.3/16/10</td>
<td></td>
<td>65/179/92/0</td>
<td>318</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>FrodoKEM-1344</td>
<td>1.4/15/6</td>
<td></td>
<td>41/109/80/0</td>
<td>351</td>
<td>3</td>
<td>-</td>
</tr>
<tr>
<td>This Work</td>
<td>SEAL-128</td>
<td>3.19/128/41</td>
<td>Virtex-7</td>
<td>654/2347/581/0</td>
<td>152</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td></td>
<td>SEAL-192</td>
<td>3.19/192/51</td>
<td></td>
<td>993/3620/843/0</td>
<td>122</td>
<td>2</td>
<td>-</td>
</tr>
<tr>
<td></td>
<td>SEAL-256</td>
<td>3.19/256/60</td>
<td></td>
<td>845/4845/1103/0</td>
<td>102</td>
<td>2</td>
<td>-</td>
</tr>
<tr>
<td>This Work</td>
<td>FALCON-I</td>
<td>2.53/18</td>
<td>Virtex-7</td>
<td>184/627/248/0</td>
<td>227</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td></td>
<td>FALCON-II</td>
<td>$\sqrt{2}/200/37$</td>
<td></td>
<td>626/2142/849/0</td>
<td>116</td>
<td>2</td>
<td>-</td>
</tr>
<tr>
<td>HW [17]</td>
<td></td>
<td>4.41/112/55</td>
<td>Spartan-6</td>
<td>122/426/123/1</td>
<td>102</td>
<td>8</td>
<td>5.01×</td>
</tr>
<tr>
<td>HW [18]</td>
<td></td>
<td>4.41/112/55</td>
<td>Spartan-6</td>
<td>150/463/45/0</td>
<td>80</td>
<td>30</td>
<td>15.69×</td>
</tr>
<tr>
<td>This Work$^a$</td>
<td></td>
<td>4.41/112/55</td>
<td>Spartan-6</td>
<td>298/970/549/0</td>
<td>263</td>
<td>3</td>
<td>-</td>
</tr>
</tbody>
</table>

Implementation Security

New applications (e.g. IoT) expose hardware to direct physical attacks / tampering: breaks crypto / key stolen

Secret Key

Plaintext → Cryptographic Hardware → Ciphertext

EM Emissions

Power Draw

Fault Attacks

Reverse Engineering & Optical Readout

0000
1101
0101
0011
Physical Side-Channel Analysis

This talk: Power and EM

Fundamental property of CMOS:
+ More practical (low-cost) than optical leakage
+ More precise than thermal leakage

\[ P_{\text{cpu}} = P_{\text{dyn}} + P_{\text{stat}} = P_{\text{trans}} + P_{\text{sc}} + P_{\text{leak}} \]

\[ P_{\text{trans}} = C_L V_{DD}^2 \cdot f \cdot P_{0\rightarrow1} \]
Side-Channel Security

Physical source: power, EM, acoustic, photonic, thermal, …
Digital source: time, micro-architectural state, memory patterns, …

Differential Power Analysis

Paul Kocher, Joshua Jaffe, and Benjamin Jun
Cryptography Research, Inc.
607 Market Street, 5th Floor
San Francisco, CA 94105, USA.
http://www.cryptography.com
E-mail: {paul, josh, ben}@cryptography.com.

Abstract. Cryptosystem designers frequently assume that secrets will be manipulated in closed, reliable computing environments. Unfortunately, actual computers and microchips leak information about the operations they process. This paper examines specific methods for analyzing power consumption measurements to find secret keys from tamper resistant devices. We also discuss approaches for building cryptosystems that can operate securely in existing hardware that leaks information.

Keywords: differential power analysis, DPA, SPA, cryptanalysis, DES

CRYPTO’99*

*Omitting TEMPEST for simplicity
FALCON’s Side-Channel Vulnerability

Key generation sub-routine leaks secret key bit values

```c
static inline uint32_t
mq_conv_small(int x)
{
    uint32_t y;
    y = (uint32_t)x;
    y += Q & -(y >> 31);
    return y;
}
```

00000000  FFFFFFFFFF
NTRU and NTRU Prime Side-Channel Vulnerability

Listing 1. NTRU Sorting Reference Implementation

```c
void crypto_sort_int32(  
    int32 *array, size_t n)  
{  
    ...  
    for (p = top; p >= 1; p >>= 1){  
        i = 0;  
        while (i + 2 * p <= n){  
            for (j = i; j < i + p; ++j){  
                int32_MINMAX(x[j], x[j+p])  
            }  
            i += 2 * p;  
        }  
    }  
    ...  
}
```

Listing 2. Implementation of int32_MINMAX

```c
#define int32_MINMAX(a,b) \  
    do { \  
        int32_t ab = (b) ^ (a); \  
        int32_t c = \  
            (int32_t)((int64_t)(b) \  
                - (int64_t)(a)); \  
        c ^= ab & (c ^ (b)); \  
        c >>= 31; \  
        c &= ab; \  
        (a) ^= c; \  
        (b) ^= c; \  
    } while(0)
```

SamplePerm Full Power Trace and Extraction

```
#define int32_MINMAX(a,b) \ 
  do { \ 
    int32_t ab = (b) ^ (a); \ 
    int32_t c = \ 
      (int32_t)((int64_t)(b) \ 
        - (int64_t)(a)); \ 
    c ^= ab & (c ^ (b)); \ 
    c >>= 31; \ 
    c &= ab; \ 
    (a) ^= c; \ 
    (b) ^= c; \ 
  } while(0)
```

Dilithium Sampling Leakage

Listing 3. Dilithium Polynomial Generation Reference Implementation

```c
void poly_challenge(poly *c,
    const uint8_t seed[SEEDBYTES])
{
    ... 
    for(i = 0; i < 8; ++i)
        signs |= (uint64_t)buf[i] << 8*i;
    pos = 8;
    for(i = 0; i < N; ++i)
        c->coeffs[i] = 0;
    for(i = N-NAU; i < N; ++i) {
        if(pos >= SHAKE256_RATE) {
            shake256_squeezeblocks(buf, 1,
                &state);
            pos = 0;
        }
        b = buf[pos++];
    } while(b > i);
    c->coeffs[i] = c->coeffs[b];
    c->coeffs[b] = (1 - 2*(signs & 1));
    signs >>= 1;
}
```

Requirements For A Differential Side-Channel Attack

An intermediate computation:
1) that combines a known value and a secret key and
2) the known value varies (i.e., not fixed)

<table>
<thead>
<tr>
<th>Input</th>
<th>Key Hypothesis</th>
<th>Power (µW)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Key=00 state</td>
<td>Key=01 state</td>
</tr>
<tr>
<td></td>
<td>$P_m$</td>
<td>$P_m$</td>
</tr>
<tr>
<td>$I_1=01$</td>
<td>01 1</td>
<td>00 0</td>
</tr>
<tr>
<td>$I_2=0f$</td>
<td>0f 4</td>
<td>0e 3</td>
</tr>
<tr>
<td>$I_{10000}=f1$</td>
<td>f1 5</td>
<td>f0 4</td>
</tr>
</tbody>
</table>
Single-Trace Differential Attacks on FrodoKEM Matrix Multiplication

- Attacker limited to a single power measurement trace
- Matrix multiplication has “multiple” intermediate computations on the same secret
  - Up to 1344 distinct computations on the same secret (S) coefficient
  - Attack splits measurements into “sub-traces” for profiling and test

\[
\begin{array}{c|c|c|c}
\text{public (A)} & \text{secret (S)} & \text{Partial Products} & \text{Result} \\
\hline
26 & 1 & 26 & 135 \\
17 & 4 & 68 & 229 \\
45 & 3 & 360 & 66 \\
\hline
14 & 4 & 14 & 135 \\
12 & 4 & 68 & 229 \\
\hline
38 & 3 & 38 & 3 \\
80 & 3 & 320 & 361 \\
1 & 1 & 36 & 361 \\
\end{array}
\]
Atacking the FALCON Signatures with Differential Power Analysis

NTRU equation:
\[ fG - gF = q \]

Public Key:
\[ h = gf^{-1} \]

If we know either polynomial ‘g’ or ‘f’, we can recover the other secret polynomial.

Attack target: Multiplication of known polynomial ‘c’ and secret polynomial ‘f’

Algorithm 2 FALCON Signature Generation Algorithm [5]

\[
\begin{align*}
\text{Input:} & \quad \text{a message } m, \text{ a secret key } sk, \text{ a bound } \beta^2 \\
\text{Output:} & \quad \text{a signature } \text{sig of } m \\
1: & \quad r \leftarrow \{0,1\}^{320} \text{ uniformly} \\
2: & \quad c \leftarrow \text{HashToPoint}(r||m) \\
3: & \quad t \leftarrow (\frac{-1}{q} FFT(c) \odot FFT(F), \frac{1}{q} FFT(c) \odot FFT(f)) \\
4: & \quad \text{do} \\
5: & \quad \text{do} \\
6: & \quad z \leftarrow \text{ffSampling}(t, T) \\
7: & \quad s \leftarrow (t - z) \begin{bmatrix} FFT(g) & -FFT(f) \\ FFT(G) & -FFT(F) \end{bmatrix} \\
8: & \quad \text{while } s^2 > [\beta^2] \\
9: & \quad (s_1, s_2) \leftarrow \text{invFFT}(s) \\
10: & \quad s \leftarrow \text{Compress}(s_2, 8 \cdot \text{sbytelen} - 328) \\
11: & \quad \text{while } s = \perp \\
12: & \quad \text{return } \text{sig} = (r, s)
\end{align*}
\]
FALCON FFT and Multiplication

Secret coefficients of $f$ can be recovered by targeting the FFT-domain multiplication.
Challenge of Attacking Multiplication

Additions remove false positives: apply extend-and-prune!
Evaluation Results

1k measurements can extract sign, 100 traces can extract exponent and mantissa

Attacking the Sign-bit
Attacking the Exponent
Attacking the Mantissa Multiplication
Attacking the Mantissa Addition
Side-Channel Security

**Masking**
- Randomize “intermediate” computations
  - Provably secure
  - Needs tuning for each $f$

**Hiding**
- Design constant power circuits
  - Automation friendly
  - Patented* and may leak

*Patent pending.
Masking Cryptographic Hardware Is Hard!

Three Takeaways

1. Quantum-secure cryptography is unavoidable

2. (Lattice-based) quantum-secure cryptography is fundamentally different

3. Need new hardware designs:
   - Optimize components
   - Design full system and explore trade-offs & design space
   - Support hybrid schemes
   - Add “implementation” security
Questions

Research Sponsors

Interested? Email: aaysu@ncsu.edu
ChatGPT Answers

When should we start using post-quantum cryptography?

It is recommended to start using post-quantum cryptography as soon as possible, even though the immediate threat of a quantum computer breaking current encryption methods is low. The reason for this is that it can take a long time to phase in a new cryptosystem and make sure it is secure and properly implemented. Additionally, as quantum computers continue to improve, the threat of them breaking current encryption methods will increase, so it is better to start preparing for this eventuality now.

It is important to note that post-quantum cryptography should not be used alone, but should be used in conjunction with currently used cryptographic methods. This will provide an extra layer of security, making it more difficult for an attacker to break the encryption even if they have access to a quantum computer.

It is also important to use post-quantum cryptography for sensitive information that needs to be protected for a long period of time, such as financial information, personal identification, and classified data.