To BLISS-B or not to be - Attacking strongSwan's Implementation of Post-Quantum Signatures

Peter Pessl[1], Leon Groot Bruinderink[2], Yuval Yarom [3]
[1] Graz University of Technology, [2] Technische Universiteit Eindhoven,
[3] University of Adelaide and Data61

CCS 2017, November 2nd

- PQ crypto is gaining a lot of traction. . .
  - NIST call, first real-world tests, efficient schemes and implementations
  - BLISS - lattice-based signatures

- But what about implementation security?
  - first works on BLISS (and lattice-based cryptography)
  - . . . but often not done in a realistic setting
  - . . . and not applicable to improved BLISS-B

Our contribution

- New side-channel key-recovery algorithm for BLISS
  - applicable to the improved BLISS-B variant
- First practical cache attack on BLISS
  - production-grade BLISS-B implementation of strongSwan VPN suite
  - 6 000 signatures for full signing-key recovery

BLISS - Lattice Signatures [DDLL13, Duc14]

- **BLISS** - Bimodal Lattice Signature Scheme [DDLL13]
- Discrete Gaussians $D_\sigma(x) \rightarrow$ dedicated samplers
- Works over ring $\mathcal{R}_q = \mathbb{Z}_q[x]/\langle x^n + 1 \rangle$, $n = 512$
  - polynomials **a**,**b**, $\mathbf{ab} = \mathbf{Ab}$, nega-cyclic rotations

$$\mathbf{A} = \begin{bmatrix} a_0 & -a_{n-1} & \cdots & -a_1 \\ a_1 & a_0 & \cdots & -a_2 \\ \vdots & \vdots & \ddots & \vdots \\ a_{n-1} & a_{n-2} & \cdots & a_0 \end{bmatrix} = \begin{bmatrix} - & \mathbf{a}_0 & - \\ - & \mathbf{a}_1 & - \\ & \vdots & \\ - & \mathbf{a}_{n-1} & - \end{bmatrix}$$

BLISS Keys

- Key generation:
  1: $\mathbf{f}, \mathbf{g} \leftarrow \{0, \pm 1, \pm 2\}^n$ (Depending on parameter set)
  2: Private key $(\mathbf{s}_1, \mathbf{s}_2) = (\mathbf{f}, 2\mathbf{g} + 1)$
  3: Public key $\mathbf{a}_q = \mathbf{s}_2 / \mathbf{s}_1 \mod q$
- BLISS-I, II: $\mathbf{f}, \mathbf{g} \leftarrow \{0, \pm 1\}^n$

BLISS - Lattice Signatures [DDLL13]

Input: Message μ, public key $\mathbf{a}_1$, private key $(\mathbf{s}_1, \mathbf{s}_2)$

Output: A signature $(\mathbf{z}_1, \mathbf{z}_2, \mathbf{c})$

1: $\mathbf{y}_1 \leftarrow D_\sigma^n$, $\mathbf{y}_2 \leftarrow D_\sigma^n$

2: $\mathbf{c} = \mathrm{H}(\mathbf{a}_1\mathbf{y}_1 + \mathbf{y}_2 || \mu)$ //binary, sparse vector

3: $(\mathbf{v}_1, \mathbf{v}_2) = (\mathbf{s}_1, \mathbf{s}_2)\mathbf{c}$

4: Sample a uniformly random bit b

5: $(\mathbf{z}_1, \mathbf{z}_2) = (\mathbf{y}_1, \mathbf{y}_2) + (-1)^b(\mathbf{v}_1, \mathbf{v}_2)$

6: Continue with some probability $f((\mathbf{s}_1, \mathbf{s}_2)\mathbf{c}, \mathbf{z})$, restart otherwise

7: **return** $(\mathbf{z}_1, \mathbf{z}_2, \mathbf{c})$

BLISS and BLISS-B [DDLL13, Duc14]

- BLISS-B $\rightarrow$ lower rejection rate, default in strongSwan
- GreedySC
  - $(\mathbf{v}_1, \mathbf{v}_2) = (\mathbf{s}_1, \mathbf{s}_2)\mathbf{c}'$, with $\mathbf{c}' \in \{-1, 0, +1\}^n$, $\mathbf{c}' \equiv \mathbf{c} \mod 2$
  - $\mathbf{c}'$ is kept secret

BLISS	BLISS-B
3: $(\mathbf{v}_1, \mathbf{v}_2) = (\mathbf{s}_1, \mathbf{s}_2)\mathbf{c}$	3: $(\mathbf{v}_1, \mathbf{v}_2) = \mathrm{GreedySC}((\mathbf{s}_1, \mathbf{s}_2), \mathbf{c})$
4: Sample a uniformly random bit b	4: Sample a uniformly random bit b
5: $(\mathbf{z}_1, \mathbf{z}_2) = (\mathbf{y}_1, \mathbf{y}_2) + (-1)^b(\mathbf{v}_1, \mathbf{v}_2)$	5: $(\mathbf{z}_1, \mathbf{z}_2) = (\mathbf{y}_1, \mathbf{y}_2) + (-1)^b(\mathbf{v}_1, \mathbf{v}_2)$

A Cache Attack on BLISS [GBHLY16]

- Cache attack on Gaussian sampler
  - partial recovery of the noise vector $\mathbf{y}_1$
- Equation $\mathbf{z}_1 = \mathbf{y}_1 + (-1)^b \mathbf{s}_1 \mathbf{c}$

$$\begin{bmatrix} \vdots \\ z_i \\ \vdots \end{bmatrix} = \begin{bmatrix} \vdots \\ y_i \\ \vdots \end{bmatrix} + (-1)^b \begin{bmatrix} & \vdots & \\ - & \mathbf{c}_i & - \\ & \vdots & \end{bmatrix} \begin{bmatrix} s_0 \\ \vdots \\ s_{n-1} \end{bmatrix}$$

$$(z_i - y_i)(-1)^b = \langle \mathbf{c}_i, \mathbf{s}_1 \rangle$$

- Filter for $z_i = y_i$

A Cache Attack on BLISS [GBHLY16]

- Gather $n = 512$ equations

$$\begin{bmatrix} - & (\mathbf{c}_i)_0 & - \\ & \vdots & \\ - & (\mathbf{c}_i)_{n-1} & - \end{bmatrix} \begin{bmatrix} s_0 \\ \vdots \\ s_{n-1} \end{bmatrix} = \begin{bmatrix} 0 \\ \vdots \\ 0 \end{bmatrix}$$

- Solve system

Limitations of the Cache Attack

- Target *research-oriented* BLISS reference implementation
  - ...and modify code, synchronized attacker
- Not applicable to BLISS-B
  - same as other works [Pes16, BBK16, EFGT16]

BLISS

$$\begin{bmatrix} 0 & 1 & \cdots & 0 \\ 0 & 0 & \cdots & -1 \\ \vdots & \vdots & \ddots & \vdots \\ 1 & 0 & \cdots & -1 \end{bmatrix} \begin{bmatrix} s_0 \\ s_1 \\ \vdots \\ s_{n-1} \end{bmatrix} = \mathbf{0}$$

BLISS-B

$$\begin{bmatrix} 0 & \pm 1 & \cdots & 0 \\ 0 & 0 & \cdots & \pm 1 \\ \vdots & \vdots & \ddots & \vdots \\ \pm 1 & 0 & \cdots & \pm 1 \end{bmatrix} \begin{bmatrix} s_0 \\ s_1 \\ \vdots \\ s_{n-1} \end{bmatrix} = \mathbf{0}$$

A New Side-Channel Key-Recovery Attack

Step 1: Gathering Samples

- Use side-channels to gather noise samples y
  - cache attack, power analysis, ...
- Collect equations

$$\begin{bmatrix} 0 & \pm 1 & \cdots & 0 \\ 0 & 0 & \cdots & \pm 1 \\ \vdots & \vdots & \ddots & \vdots \\ \pm 1 & 0 & \cdots & \pm 1 \end{bmatrix} \begin{bmatrix} s_0 \\ s_1 \\ \vdots \\ s_{n-1} \end{bmatrix} = \begin{bmatrix} 2 \\ 0 \\ \vdots \\ -3 \end{bmatrix}$$

Step 2: Finding $\mathbf{s}_1$ mod 2

- In GF(2): $-1 \equiv 1 \bmod 2$
- Solve system $\rightarrow$ $\mathbf{s}_1$ mod 2
  - LSB of the coefficients
  - BLISS-I, II $\rightarrow$ $|\mathbf{s}_1|$

$$\mathbf{s}_1 = \begin{bmatrix} 0 \\ \pm 1 \\ 0 \\ \vdots \\ \pm 1 \end{bmatrix}$$

Step 2: Correcting Errors

- Side-channels can have errors: approximate eqs.
- Solving a noisy linear system in GF(2)
  - Learning Parity with Noise (LPN)
- Our approach
  - solving LPN by decoding a random linear code
  - utilize differing error probabilities [PM16]

$$\mathbf{s}_1 = \begin{bmatrix} 0 \\ \pm 1 \\ 0 \\ \vdots \\ \pm 1 \end{bmatrix}$$

Step 3: Recovery of Twos

- BLISS-III, BLISS-IV: $\mathbf{s}_1 \in \{0, \pm 1, \pm 2\}^n$
- Use sparsity of $\mathbf{c}'$ in $\langle \mathbf{s}_1, \mathbf{c}'_i \rangle$
- Method 1: Integer Programming
  - ($|\langle \mathbf{s}_1, \mathbf{c}'_i \rangle| >$ # indexed 1s) $\rightarrow$ must be a 2 involved
- Method 2: Statistical Approach
  - ($|\langle \mathbf{s}_1, \mathbf{c}'_i \rangle|$ is large) $\rightarrow$ likely a 2 involved

$$\mathbf{s}_1 = \begin{bmatrix} 0 \\ \pm 1 \\ \pm 2 \\ \vdots \\ \pm 1 \end{bmatrix}$$

Step 4: Lattice Reduction

- Combine recovered information $|\mathbf{s}_1|$ with public key
- Public key: $\mathbf{a}_q \mathbf{s}_1 = \mathbf{s}_2$
  - $\mathbf{s}_2$: *short* vector in lattice spanned by $\mathbf{a}_q$
  - reduce lattice rank by discarding columns

$$\begin{bmatrix} a_0 & -a_{n-1} & -a_{n-2} & \cdots & -a_1 \\ a_1 & a_0 & -a_{n-1} & \cdots & -a_2 \\ a_2 & a_1 & a_0 & \cdots & -a_3 \\ \vdots & \vdots & \ddots & \vdots & \\ a_{n-1} & a_{n-2} & a_{n-2} & \cdots & a_0 \end{bmatrix} \begin{bmatrix} 0 \\ \pm 1 \\ 0 \\ \vdots \\ \pm 1 \end{bmatrix} = \mathbf{s}_2$$

Step 4: Lattice Reduction

- Reduce lattice dimension ($d = 250$)
- Solve SVP with BKZ lattice reduction
- Linear algebra to get $(\mathbf{s}_1, \mathbf{s}_2)$

Full key recovered!

Attacking strongSwans BLISS-B

Attack Target

- *Bernoulli* rejection sampling by [DDLL13]
  - bit-scanning of input x in subroutine

Sampling a bit from $\mathcal{B}(\exp(-x/(2\sigma^2)))$ for $x \in [0, 2^\ell)$

Input: $x \in [0, 2^\ell)$ an integer in binary form $x = x_{\ell-1} \dots x_0$. Precomputed table E

Output: A bit b from $\mathcal{B}(\exp(-x/(2\sigma^2)))$

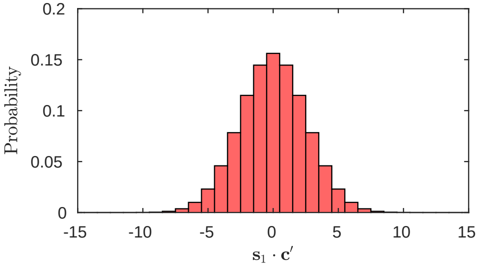
```
1: for i = ℓ − 1 downto 0 do
2:     if x_i = 1 then
3:         sample bit A_i from B(E[i])
4:         if A_i = 0 then return 0
5: return 1
```

Cache Attack

- Detect if branch $x_i = 1$ is taken at least once
  - if NOT: $x = 0 \rightarrow y = 254 \cdot \mathbb{Z}$
- Flush+Reload Cache Attack [YF14]
  - with performance degradation [ABF$^+$16]

Resynchronization

- Attack is *asynchronous*
  - need correct index
- Resynchronization
  - sampling time $\sim$ index
  - $\mathbf{s}_1\mathbf{c}'$ is small $\rightarrow z \approx 254 \cdot \mathbb{Z}$



Results

- Step 1: gathering samples
  - observe 6 000 signature generations with strongSwan
- Step 2: $\mathbf{s}_1 \bmod 2$
  - 98% success rate, avg. runtime $\approx$ 1 minute (64 threads)
- Step 3: Recovering 2s
  - ...not needed, focus on BLISS-I for strongSwan tests
- Step 4: lattice reduction
  - BLISS-I: always successful, avg. runtime 4-5 minutes

What can we do?

Countermeasures

- Shuffling the noise vector
  - also has flaws [Pes16]
- Constant-time samplers
  - difficult to implement, still vulnerable to power analysis
- Don't use Gaussians!
  - Gaussians are: difficult to implement, extremely prone to SCA
  - replace with, e.g., uniform distribution (Dilithium [DLL$^+$17])

SCIENCE ▪ PASSION ▪ TECHNOLOGY

To BLISS-B or not to be - Attacking strongSwan's Implementation of Post-Quantum Signatures

Peter Pessl[1], Leon Groot Bruinderink[2], Yuval Yarom [3]

[1] Graz University of Technology, [2] Technische Universiteit Eindhoven, [3] University of Adelaide and Data61

CCS 2017, November 2nd

Bibliography I

[ABF+16] Thomas Allan, Billy Bob Brumley, Katrina E. Falkner, Joop van de Pol, and Yuval Yarom. Amplifying side channels through performance degradation. In *ACSAC 2016*, pages 422–435, 2016.

[BBK16] Nina Bindel, Johannes A. Buchmann, and Juliane Krämer. Lattice-based signature schemes and their sensitivity to fault attacks. In *FDTC 2016*, pages 63–77, 2016.

[DDLL13] Léo Ducas, Alain Durmus, Tancrède Lepoint, and Vadim Lyubashevsky. Lattice signatures and bimodal gaussians. In *CRYPTO 2013*, pages 40–56, 2013.

[DLL+17] Léo Ducas, Tancrède Lepoint, Vadim Lyubashevsky, Peter Schwabe, Gregor Seiler, and Damien Stehlé. CRYSTALS - dilithium: Digital signatures from module lattices. 2017.

[Duc14] Léo Ducas. Accelerating bliss: the geometry of ternary polynomials. 2014.

[EFGT16] Thomas Espitau, Pierre-Alain Fouque, Benoît Gérard, and Mehdi Tibouchi. Loop abort faults on lattice-based fiat-shamir & hash'n sign signatures. 2016.

Bibliography II

[GBHLY16] Leon Groot Bruinderink, Andreas Hülsing, Tanja Lange, and Yuval Yarom. Flush, gauss, and reload - A cache attack on the BLISS lattice-based signature scheme. In *CHES 2016*, pages 323–345, 2016. Full version available at: `http://ia.cr/2016/300`.

[Pes16] Peter Pessl. Analyzing the shuffling side-channel countermeasure for lattice-based signatures. In *INDOCRYPT 2016*, pages 153–170, 2016.

[PM16] Peter Pessl and Stefan Mangard. Enhancing side-channel analysis of binary-field multiplication with bit reliability. In *CT-RSA 2016*, pages 255–270, 2016.

[YF14] Yuval Yarom and Katrina Falkner. FLUSH+RELOAD: A high resolution, low noise, L3 cache side-channel attack. In *23rd USENIX Security Symposium*, pages 719–732, 2014.