Generalized Howgrave-Graham–Szydlo and Side-Channel Attacks against BLISS

Mehdi Tibouchi

NTT Secure Platform Laboratories

AGC^{2}T, 2017–06–22

joint work with T. Espitau, P.-A. Fouque and B. Gérard

Outline

Towards postquantum cryptography

- Quantum computers would break all currently deployed public-key crypto: RSA, discrete logs, elliptic curves
- Agencies warn that we should prepare the transition to quantum-resistant crypto
  - NSA deprecating Suite B (elliptic curves)
  - NIST starting their postquantum competition
- In theory, plenty of known cryptosystems are quantum-resistant
  - Some primitives achieved with codes, hash trees, multivariate crypto, knapsacks, isogenies...
  - Almost everything possible with lattices
- In practice, few actual implementations
  - Secure parameters often unclear
  - Concrete software/hardware implementation papers quite rare
  - Almost no consideration for implementation attacks
- Serious issue if we want practical postquantum crypto

Lattice-based cryptography

- Roughly speaking, lattice-based cryptography is crypto based on hard problems in the "geometry of numbers"
  - given a basis of a submodule M of large rank in $\mathbb{Z}^m$, find a short element of M (for the Euclidean norm)
- Very fruitful class of problems for constructing interesting crypto; believed to remain hard even against quantum computers
- Drawback: for security, dimensions in the hundreds or thousands are necessary, resulting in large keys and limited performance
- Solution: use modules over larger rings, e.g. ideals of $\mathscr{O}_K$ for K number field of large degree over $\mathbb{Q}$
  - in practice, people use $K = \mathbb{Q}(\zeta_m)$, $m = 2^k$
  - interesting playground for algorithmic number theorists (cf. Alice Silverberg's talk)

Attacks on lattice-based cryptography

- Nice feature of lattice-based cryptosystems: they usually come with very strong security arguments
  - "if you can break this scheme, you can solve lattice problems of the same dimension (and over the same ring) in the worst case"
- Thus, to attack a lattice-based cryptosystem "algorithmically", you have to make significant progress on the analysis of a number-theoretic problem believed to be hard
- Some suspect this may be feasible over certain rings (e.g. in cyclotomic fields); research is ongoing
- However, this is not what this talk is about (too hard for me). This talk is about cheating to break things!

Black-box vs real-world security

- Consider the security of e.g. digital signatures
- Traditional, “black-box” view of security:
  - the attacker, Alice, interacts with the signer, Bob
  - Alice sends Bob messages to sign, only gets the results of Bob’s computation (no other info about the computation is revealed)
  - based on that, Alice tries to forge new signatures/extract info about Bob’s signing key
- Real-world security:
  - Bob is actually a smart card, say
  - Alice can measure all sorts of emanation from the card as it operates, or mess with it in various ways
  - all that extra information can be useful to break things!

Implementation attacks

- The security guarantees offered by “security proofs” for lattice-based crypto are in the black-box model
- But to break a real-world crypto implementation, no need to play by the rules of that model
- This talk: measure the side-channel leakage of an implementation of lattice-based signatures, and use it together with a little bit of number theory to recover the entire key
  - specifically, electromagnetic emanations
  - it would also work with power consumption, etc.
- FWIW: there are other types of interesting implementation attacks, including fault attacks (actively tamper with the device during the computation) that also lead to key recovery (but with even less math involved)

Outline

BLISS: the basics

- The BLISS signature scheme is one of the top contenders for postquantum signatures
- Introduced by Ducas, Durmus, Lepoint and Lyubashevsky at CRYPTO'13
- Implementations exist on various platforms: desktop computers, microcontrollers/smartcards, and even hardware (FPGAs)
- Deployed in the VPN library strongSwan
- Based on ideal lattices over the ring of cyclotomic integers $R = \mathbb{Z}[\zeta]$ with ζ primitive 1024-th root of unity

BLISS: signing and verification keys

- We identify R as $\mathbb{Z}[\mathbf{x}]/(\mathbf{x}^n + 1)$ (with $n = 512$), and elements of R as polynomials in $\mathbf{x}$ of degree < 512, or vectors in $\mathbb{Z}^{512}$
- We fix q a rational prime which splits completely in R ($q = 12289$)
- The secret signing key consists of two random elements $\mathbf{s}_1, \mathbf{s}_2 \in R$ with coefficients in $\{-1, 0, 1\}$, sparse
- The verification key is $\mathbf{a} = -\mathbf{s}_2/\mathbf{s}_1 \bmod q$
  - restart if $\mathbf{s}_1$ not invertible

BLISS: signature

function $\textsc{Sign}(\mu, pk = \mathbf{a},\ sk = \mathbf{S} = (\mathbf{s}_1, \mathbf{s}_2))$

  $\mathbf{y}_1, \mathbf{y}_2 \leftarrow D^n_{\mathbb{Z},\sigma}$ ▷ Gaussian sampling

  $\mathbf{c} \leftarrow H(\mathbf{a} \cdot \mathbf{y}_1 + \mathbf{y}_2, \mu)$ ▷ special hashing

  choose a random bit b

  $\mathbf{z}_1 \leftarrow \mathbf{y}_1 + (-1)^b \mathbf{s}_1 \mathbf{c}$

  $\mathbf{z}_2 \leftarrow \mathbf{y}_2 + (-1)^b \mathbf{s}_2 \mathbf{c}$

  continue with probability $1/\big(M \exp(-\|\mathbf{Sc}\|^2/(2\sigma^2)) \cosh(\langle \mathbf{z}, \mathbf{Sc} \rangle/\sigma^2\big)$ otherwise **restart**

  $\mathbf{z}_2^\dagger \leftarrow \textsc{Compress}(\mathbf{z}_2)$

  return $(\mathbf{z}_1, \mathbf{z}_2^\dagger, \mathbf{c})$

end function

BLISS: signature

function $\textsc{Sign}(\mu, pk = \mathbf{a},\ sk = \mathbf{S} = (\mathbf{s}_1, \mathbf{s}_2))$
  $\mathbf{y}_1, \mathbf{y}_2 \leftarrow D^n_{\mathbb{Z},\sigma}$ $\triangleright$ Gaussian sampling
  $\mathbf{c} \leftarrow H(\mathbf{a} \cdot \mathbf{y}_1 + \mathbf{y}_2, \mu)$ $\triangleright$ special hashing
  choose a random bit b
  $\mathbf{z}_1 \leftarrow \mathbf{y}_1 + (-1)^b \mathbf{s}_1 \mathbf{c}$
  $\mathbf{z}_2 \leftarrow \mathbf{y}_2 + (-1)^b \mathbf{s}_2 \mathbf{c}$
  continue with probability $1/\big(M \exp(-\|\mathbf{Sc}\|^2/(2\sigma^2)) \cosh(\langle \mathbf{z}, \mathbf{Sc} \rangle/\sigma^2\big)$ otherwise **restart**
  $\mathbf{z}_2^\dagger \leftarrow \textsc{Compress}(\mathbf{z}_2)$
  return $(\mathbf{z}_1, \mathbf{z}_2^\dagger, \mathbf{c})$
end function

BLISS: signature

function $\textsc{Sign}(\mu, pk = \mathbf{a},\ sk = \mathbf{S} = (\mathbf{s}_1, \mathbf{s}_2))$

  $\mathbf{y}_1, \mathbf{y}_2 \leftarrow D^n_{\mathbb{Z},\sigma}$ ▷ Gaussian sampling

  $\mathbf{c} \leftarrow H(\mathbf{a} \cdot \mathbf{y}_1 + \mathbf{y}_2, \mu)$ ▷ special hashing

  choose a random bit b

  $\mathbf{z}_1 \leftarrow \mathbf{y}_1 + (-1)^b \mathbf{s}_1 \mathbf{c}$

  $\mathbf{z}_2 \leftarrow \mathbf{y}_2 + (-1)^b \mathbf{s}_2 \mathbf{c}$

  continue with probability $1/\left(M \exp(-\|\mathbf{Sc}\|^2/(2\sigma^2)) \cosh(\langle \mathbf{z}, \mathbf{Sc}\rangle/\sigma^2\right)$ otherwise **restart**

  $\mathbf{z}_2^\dagger \leftarrow \textsc{Compress}(\mathbf{z}_2)$

  return $(\mathbf{z}_1, \mathbf{z}_2^\dagger, \mathbf{c})$

end function

BLISS: signature

function $\textsc{Sign}(\mu, pk = \mathbf{a},\ sk = \mathbf{S} = (\mathbf{s}_1, \mathbf{s}_2))$

  $\mathbf{y}_1, \mathbf{y}_2 \leftarrow D^n_{\mathbb{Z},\sigma}$ ▷ Gaussian sampling

  $\mathbf{c} \leftarrow H(\mathbf{a} \cdot \mathbf{y}_1 + \mathbf{y}_2, \mu)$ ▷ special hashing

  choose a random bit b

  $\mathbf{z}_1 \leftarrow \mathbf{y}_1 + (-1)^b \mathbf{s}_1 \mathbf{c}$

  $\mathbf{z}_2 \leftarrow \mathbf{y}_2 + (-1)^b \mathbf{s}_2 \mathbf{c}$

  continue with probability
$1/\big(M \exp(-\|\mathbf{Sc}\|^2/(2\sigma^2)) \cosh(\langle \mathbf{z}, \mathbf{Sc}\rangle/\sigma^2\big)$ otherwise **restart**

  $\mathbf{z}_2^\dagger \leftarrow \textsc{Compress}(\mathbf{z}_2)$

  return $(\mathbf{z}_1, \mathbf{z}_2^\dagger, \mathbf{c})$

end function

BLISS: signature

function $\textsc{Sign}(\mu, pk = \mathbf{a},\ sk = \mathbf{S} = (\mathbf{s}_1, \mathbf{s}_2))$

  $\mathbf{y}_1, \mathbf{y}_2 \leftarrow D^n_{\mathbb{Z},\sigma}$ ▷ Gaussian sampling

  $\mathbf{c} \leftarrow H(\mathbf{a} \cdot \mathbf{y}_1 + \mathbf{y}_2, \mu)$ ▷ special hashing

  choose a random bit b

  $\mathbf{z}_1 \leftarrow \mathbf{y}_1 + (-1)^b \mathbf{s}_1 \mathbf{c}$

  $\mathbf{z}_2 \leftarrow \mathbf{y}_2 + (-1)^b \mathbf{s}_2 \mathbf{c}$

  continue with probability $1/\big(M \exp(-\|\mathbf{Sc}\|^2/(2\sigma^2)) \cosh(\langle \mathbf{z}, \mathbf{Sc}\rangle/\sigma^2\big)$ otherwise **restart**

  $\mathbf{z}_2^\dagger \leftarrow \textsc{Compress}(\mathbf{z}_2)$

  return $(\mathbf{z}_1, \mathbf{z}_2^\dagger, \mathbf{c})$

end function

BLISS: verification

1: **function** $\textsc{Verify}(\mu, \mathbf{a}, (\mathbf{z}_1, \mathbf{z}_2^\dagger, \mathbf{c}))$
2: $\quad \mathbf{z}_2' \leftarrow \textsc{Uncompress}(\mathbf{z}_2^\dagger)$
3: $\quad$ **if** $\|(\mathbf{z}_1 | \mathbf{z}_2')\|_2 > B_2$ **then** reject
4: $\quad$ **if** $\|(\mathbf{z}_1 | \mathbf{z}_2')\|_\infty > B_\infty$ **then** reject
5: $\quad$ **accept iff** $\mathbf{c} = H(\mathbf{a} \cdot \mathbf{z}_1 + \mathbf{z}_2', \mu)$
6: **end function**

Outline

Attack overview

- The rejection sampling step is the cornerstone of BLISS security (difference with NTRUSign) and efficient (the bimodal aspect)
- In practice: difficult to implement (needs high-precision evaluation of transcendental functions), so some tricks have to be used
- The optimized version of the rejection sampling uses iterated Bernoulli trials on each of the bits of $\|\mathbf{Sc}\|^2$; as a result, we can read that value on a power analysis/electromagnetic analysis trace
- This yields to the recovery of the relative norm $\mathbf{s}_1 \cdot \bar{\mathbf{s}_1}$ in the totally real subfield (and similarly for $\mathbf{s}_2$)
- Algorithmic number theoretic techniques (Howgrave-Graham–Szydlo) can then be used to retrieve the $\mathbf{s}_i$ up to a root of unity (which is a complete break!)

BLISS rejection sampling

function $\textsc{SampleBernExp}(x \in [0, 2^\ell) \cap \mathbb{Z})$
  for $i = 0$ to $\ell - 1$ **do**
    if $x_i = 1$ **then**
      Sample $a \leftarrow \mathscr{B}_{c_i}$
      if $a = 0$ **then return** 0
    end if
  end for
  return 1
end function $\triangleright\ x = K - \|\mathbf{Sc}\|^2$

function $\textsc{SampleBernCosh}(x)$
  Sample $a \leftarrow \mathscr{B}_{\exp(-x/f)}$
  if $a = 1$ **then return** 1
  Sample $b \leftarrow \mathscr{B}_{1/2}$
  if $b = 1$ **then restart**
  Sample $c \leftarrow \mathscr{B}_{\exp(-x/f)}$
  if $c = 1$ **then restart**
  return 0
end function $\triangleright\ x = 2 \cdot \langle \mathbf{z}, \mathbf{Sc} \rangle$

Sampling algorithms for the distributions $\mathscr{B}_{\exp(-x/f)}$ and $\mathscr{B}_{1/\cosh(x/f)}$ ($c_i = 2^i/f$ precomputed)

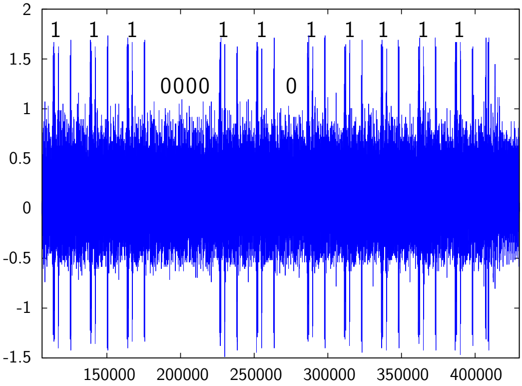
BLISS rejection sampling

function $\textsc{SampleBernExp}(x \in [0, 2^\ell) \cap \mathbb{Z})$
  for $i = 0$ to $\ell - 1$ **do**
    if $x_i = 1$ **then**
      Sample $a \leftarrow \mathscr{B}_{c_i}$
      if $a = 0$ **then return** 0
    end if
  end for
  return 1
end function ▷ $x = K - \|\mathbf{Sc}\|^2$

function $\textsc{SampleBernCosh}(x)$
  Sample $a \leftarrow \mathscr{B}_{\exp(-x/f)}$
  if $a = 1$ **then return** 1
  Sample $b \leftarrow \mathscr{B}_{1/2}$
  if $b = 1$ **then restart**
  Sample $c \leftarrow \mathscr{B}_{\exp(-x/f)}$
  if $c = 1$ **then restart**
  return 0
end function ▷ $x = 2 \cdot \langle \mathbf{z}, \mathbf{Sc} \rangle$

Sampling algorithms for the distributions $\mathscr{B}_{\exp(-x/f)}$ and $\mathscr{B}_{1/\cosh(x/f)}$ ($c_i = 2^i/f$ precomputed)

Experimental leakage



Electromagnetic measure of BLISS rejection sampling for norm $\|\mathbf{Sc}\|^2 = 14404$. One reads the value:

$$K - \|\mathbf{Sc}\|^2 = 46539 - 14404 = \overline{11100001101111}_2$$

Outline

Exploiting the leakage

- Each time a signature is computed, we obtain the Euclidean norm $\|\mathbf{s}_1 \cdot \mathbf{c}\|^2 + \|\mathbf{s}_2 \cdot \mathbf{c}\|^2$, where $\mathbf{c}$ is a known element of R that changes every time
- In other words, each signature gives a $\mathbb{Z}$-linear equation on the coefficients of the relative norms $\mathbf{s}_1 \cdot \bar{\mathbf{s}}_1$ and $\mathbf{s}_2 \cdot \bar{\mathbf{s}}_2$
- These elements are in $\mathbb{Q}(\zeta + \zeta^{-1})$ (degree 256 over $\mathbb{Q}$) so collecting around $2 \times 256 = 512$ signatures should yield a linear system of full rank, and let us recover both of the relative norms
  - the linear equations really are independent w.h.p.
  - very efficient in practice
  - the collection of 512 EM traces is an easy task by the standards of side-channel analysis
- Then, how can we use our knowledge of $\mathbf{s}_1 \cdot \bar{\mathbf{s}}_1$ and $\mathbf{s}_2 \cdot \bar{\mathbf{s}}_2$ to recover $\mathbf{s}_1$ and $\mathbf{s}_2$ themselves?
- This is where Howgrave-Graham–Szydlo comes into play

Howgrave-Graham–Szydlo (I)

- The situation is as follows: for $\mathbf{s}$ in the cyclotomic ring $\mathbb{Z}[\zeta]$, we are given the relative norm $\mathbf{r} = \mathbf{s} \cdot \bar{\mathbf{s}}$ in the totally real subfield. Can we recover $\mathbf{s}$?
- First, we compute the absolute norm:

$$N = N_{\mathbb{Q}(\zeta)/\mathbb{Q}}(\mathbf{s}) = \sqrt{N_{\mathbb{Q}(\zeta)/\mathbb{Q}}(\mathbf{r})}$$

- Suppose that $N = p$ is prime. This heuristically happens with significant probability
  - variant of the result saying that the density of ideals with prime norm among ideals of norm $< x$ is asymptotically $1/\log x$ (Landau)
  - we can bound the norm of $\mathbf{s}$, and $\mathbf{s}$ heuristically behaves like a random element of R up to that bound
  - experimentally, around 1% of keys $\mathbf{s}$ satisfy the condition

Howgrave-Graham–Szydlo (I)

- The situation is as follows: for $\mathbf{s}$ in the cyclotomic ring $\mathbb{Z}[\zeta]$, we are given the relative norm $\mathbf{r} = \mathbf{s} \cdot \bar{\mathbf{s}}$ in the totally real subfield. Can we recover $\mathbf{s}$?
- First, we compute the absolute norm:

$$N = N_{\mathbb{Q}(\zeta)/\mathbb{Q}}(\mathbf{s}) = \sqrt{N_{\mathbb{Q}(\zeta)/\mathbb{Q}}(\mathbf{r})}$$

- Suppose that $N = p$ is prime. This heuristically happens with significant probability
- We must have $p \equiv 1 \pmod 4$, because p is the norm of an element of $\mathbb{Z}[\sqrt{-1}]$
- In particular, p splits as $\pi\bar{\pi}$ in $\mathbb{Z}[\sqrt{-1}]$. Then, $\mathbf{r}R$ is the product of the two ideals $(\mathbf{r}, \pi)$ and $(\mathbf{r}, \bar{\pi})$: one of them is thus $\mathbf{s}R$ and the other is $\bar{\mathbf{s}}R$

Howgrave-Graham–Szydlo (II)

- Previous slide: when we are given $\mathbf{r} = \mathbf{s}\bar{\mathbf{s}}$ for $\mathbf{s} \in R$ of prime absolute norm, we can recover 2 possible candidates for the principal ideal $\mathbf{s}R$
- More generally, if we are able to factor the absolute norm N of $\mathbf{s}$, a similar approach yields a polynomial number of candidates for $\mathbf{s}R$
  - basically, write all the possible ways in which $\mathbf{r}R$ decomposes as a product of two conjugate ideals of norm N
- Is this sufficient to recover $\mathbf{s}$?
- Usually, finding a generator of a prime ideal is hard. However, in our case, we also have the relative norm $\mathbf{r}$ of the generator
- This is just what we need to apply a magical algorithm due to Gentry and Szydlo, which recovers the generator up to a root of unity!

Completing the attack

- To sum up, assuming that we can factor the absolute norm of $\mathbf{s}_1$, we recover a small number of candidates for $\mathbf{s}_1$, up to multiplication by a root of unity
- Checking whether a solution is correct is easy
  - compute the corresponding candidate for $\mathbf{s}_2$ as $\mathbf{a} \cdot \mathbf{s}_1 \bmod q$
  - it should have coefficients in $\{-1, 0, 1\}$ and be sparse
- Moreover, multiplying a correct key $(\mathbf{s}_1, \mathbf{s}_2)$ by a root of unity results in a completely equivalent key, so we are done!
- Attack works for weak keys for which we can factor the absolute norm of either $\mathbf{s}_1$ or $\mathbf{s}_2$: for example when the norm is of the form $N_0 p$ where N_0 is smooth (removed by trial division) and p prime

Efficiency of the attack

	n	$B = 5$	$B = 65537$	$B = 655373$	$B = 6553733$
BLISS-0	256	3%	3.8%	6%	6.5%
BLISS-I/II	512	1.5%	2%	2.8%	3.7%
BLISS-III/IV	512	1%	1.75%	2%	2.5%

Experimental density of keys with semi-smooth absolute norm ($N = N_0 \cdot p$ with B-smooth N_0) for various BLISS parameters

Field size n	32	64	128	256	512
CPU time	0.6 s	13 s	21 min.	17h 22 min.	38 days
Clock cycles	$\approx 2^{30}$	$\approx 2^{35}$	$\approx 2^{41}$	$\approx 2^{47}$	$\approx 2^{53}$

Average running time of the attack for various field sizes n

BLISS parameters: $n = 256$ or 512

Outline

Introduction

The side-channel leakage in BLISS

The BLISS signature scheme

The rejection sampling leakage

Exploiting the leakage

Applying Howgrave-Graham–Szydlo

What about the inner product leakage?

What about the inner product leakage? (I)

- Recall the rejection sampling probability of BLISS signing:

$$1\Bigg/\left(M\exp\left(-\frac{\|\mathbf{Sc}\|^2}{2\sigma^2}\right)\cosh\left(\frac{\langle \mathbf{z},\mathbf{Sc}\rangle}{\sigma^2}\right)\right),$$

- The exp part of the rejection sampling leaks $\|\mathbf{Sc}\|^2$ and ultimately the relative norm of $\mathbf{s}_1$ and $\mathbf{s}_2$: what we have used so far
- Can't we use the cosh part instead? It directly leaks:

$$\langle \mathbf{z}_1, \mathbf{s}_1\mathbf{c}\rangle + \langle \mathbf{z}_2, \mathbf{s}_2\mathbf{c}\rangle$$

- If we know $(\mathbf{c}, \mathbf{z}_1, \mathbf{z}_2)$, this gives a *linear* relation on the secret: recover everything from around 1024 signatures without breaking a sweat!

What about the inner product leakage? (II)

- Problem: signatures do not contain $\mathbf{z}_2$, but only a compressed variant $\mathbf{z}_2^\dagger$, and the compression is lossy: we only obtain a noisy linear system on the secret
- Our first reaction: this is like Learning With Errors in twice the original dimension, so probably hopeless
- Update (recent work with J. Bootle): not hopeless at all. Since there is no modular reduction, we can simply approach the problem with linear least squares
- Works on 100% of keys, but needs ≈ 30000 signatures vs. ≈ 512 for Howgrave-Graham–Szydlo

Conclusion and possible countermeasures

- Postquantum crypto in general, and lattices in particular, are hot topics
- Many constructions use some algebraic number theory, but haven't been looked at by actual mathematicians
  - you can probably find many problems in our schemes!
  - implementation attacks in particular are an easy way to wreak havoc on all this stuff
  - they have to be considered before standardization/deployment
- Possible countermeasures?
  - compute rejection probability with floating point arithmetic (slow)
  - use a constant-time Bernoulli sampling (doable)
  - prefer a scheme with simpler structure (without those pesky Gaussians!)