Physical Attacks Against Lattice-Based Schemes

Thomas Espitau

joint work with P.-A. Fouque, B. Gérard and M. Tibouchi

April 30, 2017

Outline

Outline

Breaking provable crypto is harder

- Most crypto proposed in the last 15–20 years: provably secure
- Breaking it = provably as hard as solving some algorithmic problem like integer factorization or computing discrete logs
- Hence, cryptanalysis = major algorithmic advance?

Yet, many attacks against deployed crypto

The crypto protocol that is perhaps most used in everyday life, TLS, is attacked all the time!

```
Internet Engineering Task Force (IETF)                        Y. Sheffer
Request for Comments: 7457                                      Porticor
Category: Informational                                          R. Holz
ISSN: 2070-1721                       Technische Universitaet Muenchen
                                                         P. Saint-Andre
                                                                   &yet
                                                          February 2015

      Summarizing Known Attacks on Transport Layer Security (TLS)
                        and Datagram TLS (DTLS)

Abstract

   Over the last few years, there have been several serious attacks on
   Transport Layer Security (TLS), including attacks on its most
   commonly used ciphers and modes of operation.  This document
   summarizes these attacks, with the goal of motivating generic and
   protocol-specific recommendations on the usage of TLS and Datagram
   TLS (DTLS).
```

Yet, many attacks against deployed crypto

The crypto protocol that is perhaps most used in everyday life, TLS, is attacked all the time!

Table of Contents

So how do people actually break crypto?

- Very rarely: major algorithmic improvement
  - *Biggest one recently: progress on small characteristic discrete logarithms/pairings*
- More commonly: non-provably secure schemes shown to be insecure
  - Several of the TLS attacks
  - Many legacy scheme still in use could be broken (e.g. PKCS#1v1.5 signatures?)
- Most importantly: implementation attacks!

So how do people actually break crypto?

- Very rarely: major algorithmic improvement
  - *Biggest one recently: progress on small characteristic discrete logarithms/pairings*
- More commonly: non-provably secure schemes shown to be insecure
  - Several of the TLS attacks
  - Many legacy scheme still in use could be broken (e.g. PKCS#1v1.5 signatures?)
- Most importantly: implementation attacks!

So how do people actually break crypto?

- Very rarely: major algorithmic improvement
  - *Biggest one recently: progress on small characteristic discrete logarithms/pairings*

- More commonly: non-provably secure schemes shown to be insecure
  - Several of the TLS attacks
  - Many legacy scheme still in use could be broken (e.g. PKCS#1v1.5 signatures?)

- Most importantly: implementation attacks!

Black-box vs real-world security

- Consider the security of e.g. RSA 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 leaks)
  - based on that, Alice tries to forge new signatures/extract info about Bob's signing key

Black-box vs real-world security

- Consider the security of e.g. RSA signatures

- 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

- To break a real-world crypto implementation, no need to play by the rules of black-box security

- In particular, provably secure schemes can be broken by bypassing the (usually black-box) security model
  - *Remark: some attempts to also capture non black-box attacks in security proofs (e.g. leakage-resilient crypto...)*

- These are implementation attacks

Implementation attacks

- To break a real-world crypto implementation, no need to play by the rules of black-box security

- In particular, provably secure schemes can be broken by bypassing the (usually black-box) security model
    - *Remark: some attempts to also capture non black-box attacks in security proofs (e.g. leakage-resilient crypto...)*

- These are implementation attacks

Implementation attacks

- To break a real-world crypto implementation, no need to play by the rules of black-box security

- In particular, provably secure schemes can be broken by bypassing the (usually black-box) security model
  - *Remark: some attempts to also capture non black-box attacks in security proofs (e.g. leakage-resilient crypto...)*

- These are implementation attacks

Various types of implementation attacks

- Correctness attacks: use the implementation as a black box, but send malformed/incorrect/invalid/malicious inputs
- Side-channel attacks: passive physical attacks, exploiting information leakage about the computation or the keys
- Fault attacks: active physical attacks, trying to extract secret information by tampering with the device to cause errors during the cryptographic computation

Various types of implementation attacks

- Correctness attacks: use the implementation as a black box, but send malformed/incorrect/invalid/malicious inputs

- Side-channel attacks: passive physical attacks, exploiting information leakage about the computation or the keys

- Fault attacks: active physical attacks, trying to extract secret information by tampering with the device to cause errors during the cryptographic computation

Various types of implementation attacks

- Correctness attacks: use the implementation as a black box, but send malformed/incorrect/invalid/malicious inputs
- Side-channel attacks: passive physical attacks, exploiting information leakage about the computation or the keys
- Fault attacks: active physical attacks, trying to extract secret information by tampering with the device to cause errors during the cryptographic computation

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 depreciating Suite B (elliptic curves)
  - NIST starting their postquantum competition

Towards postquantum cryptography

- In theory, plenty of known schemes are quantum-resistant
  - Some primitives achieved with codes, hash trees, multivariate crypto, knapsacks, isogenies...
  - Almost everything possible with lattices
- In practice, very 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

Towards postquantum cryptography

- In theory, plenty of known schemes are quantum-resistant
  - Some primitives achieved with codes, hash trees, multivariate crypto, knapsacks, isogenies...
  - Almost everything possible with lattices
- In practice, very 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

Implementations of lattice-based schemes (I)

- Implementation work on lattice-based crypto is limited and mostly academic

- A number of papers describing implementations of inefficient schemes
  - Encryption: implementation of Lindner–Peikert (CHES'12, plaintexts of several MBs)
  - Signatures: implementation of GPV (SAC'13, keys of dozen MBs)
  - Other primitives: a few papers about ID-schemes, homomorphic encryption, etc.

Implementations of lattice-based schemes (II)

- One scheme has "industry" backing and quite a bit of code: NTRU
  - NTRUEncrypt is an ANSI standard, and believed to be okay
  - NTRUSign is a trainwreck that has been patched and broken many times

Implementations of lattice-based schemes (II)

- In terms of practical schemes, other than NTRU, main efforts on signatures
  - GLP: improvement of Lyubashevsky signatures, efficient in SW and HW (CHES'12)
  - BLISS: improvement of GPL, even better (CRYPTO'13, CHES'14)
  - DLP: hash-and-sign scheme using GPV sampling on NTRU lattices (AC'14)
  - A few others: PASSSign (ACNS'14), TESLA (AFRICACRYPT'16), etc.

Implementation attacks on lattice-based schemes

- Survey by Taha and Eisenbarth (eprint 2015/1083) on implementation attacks against postquantum schemes; thorough literature review
- For lattice-based schemes, only referenced attacks are against NTRU
  - NTRUEncrypt: a few papers about timing attacks (CT-RSA'07), power analysis (RFIDSec'08+journals) and faults (JCEN, IEICE Trans.)
  - NTRUSign: one paper about faults (Cryptogr. and Comm.)
- Our recent work: fault attacks (SAC 2016) and side-channel analysis (to appear) on lattice-based signatures

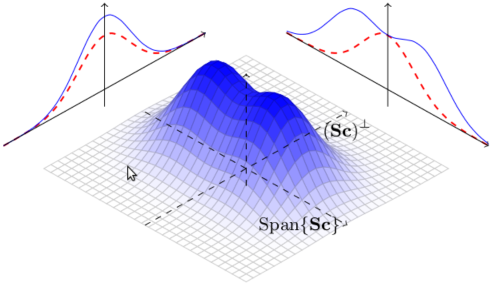
Outline

BLISS: the basics

- Introduced by Ducas, Durmus, Lepoint and Lyubashevsky at CRYPTO'13
- Improvement of the earlier Ring-SIS-based scheme of Lyubashevsky (EC'12)
- Still following the structure of "Fiat–Shamir with aborts"
- Still defined over some ring $\mathcal{R} = \mathbb{Z}[\mathbf{x}]/(\mathbf{x}^n + 1)$
- Main improvement: use bimodal Gaussian distributions to reduce the size of parameters

BLISS: the basics

- Main improvement: Reduce the size of parameters by Bimodal Gaussian distributions



BLISS: the basics

- Main improvement: Reduce the size of parameters by Bimodal Gaussian distributions

Distributio Camelus bactrianus

BLISS: key generation

function $\textsc{KeyGen}()$
  choose $\mathbf{f}, \mathbf{g}$ as uniform polynomials with exactly $d_1 = \lceil \delta_1 n \rceil$ entries in $\{\pm 1\}$ and $d_2 = \lceil \delta_2 n \rceil$ entries in $\{\pm 2\}$
  $\mathbf{S} = (\mathbf{s}_1, \mathbf{s}_2)^T \leftarrow (\mathbf{f}, 2\mathbf{g} + 1)^T$
  if $N_\kappa(\mathbf{S}) \geq C^2 \cdot 5 \cdot (\lceil \delta_1 n \rceil + 4 \lceil \delta_2 n \rceil) \cdot \kappa$ **then restart**
  if $\mathbf{f}$ is not invertible **then restart**
  $\mathbf{a}_q = (2\mathbf{g} + 1)/\mathbf{f} \bmod q$
  return $(pk = \mathbf{A}, sk = \mathbf{S})$ where $\mathbf{A} = (\mathbf{a}_1 = 2\mathbf{a}_q, q - 2) \bmod 2q$
end function

BLISS: signature

function $\textsc{Sign}(\mu, pk = \mathbf{A},\ sk = \mathbf{S})$

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

$\quad \mathbf{u} = \zeta \cdot \mathbf{a}_1 \cdot \mathbf{y}_1 + \mathbf{y}_2 \bmod 2q$ $\triangleright$ $\zeta = 1/(q-2)$

$\quad \mathbf{c} \leftarrow H(\lfloor \mathbf{u} \rceil_d \bmod p, \mu)$ $\triangleright$ special hashing

$\quad$ choose a random bit b

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

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

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

$\quad \mathbf{z}_2^\dagger \leftarrow (\lfloor \mathbf{u} \rceil_d - \lfloor \mathbf{u} - \mathbf{z}_2 \rceil_d) \bmod p$

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

end function

BLISS: signature

function $\textsc{Sign}(\mu, pk = \mathbf{A},\ sk = \mathbf{S})$
    $\mathbf{y}_1, \mathbf{y}_2 \leftarrow D^n_{\mathbb{Z},\sigma}$ ▷ Gaussian sampling
    $\mathbf{u} = \zeta \cdot \mathbf{a}_1 \cdot \mathbf{y}_1 + \mathbf{y}_2 \bmod 2q$ ▷ $\zeta = 1/(q-2)$
    $\mathbf{c} \leftarrow H(\lfloor \mathbf{u} \rceil_d \bmod p, \mu)$ ▷ 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
$1/\Big(M \exp(-\|\mathbf{Sc}\|/(2\sigma^2)) \cosh(\langle \mathbf{z}, \mathbf{Sc}\rangle/\sigma^2\Big)$ otherwise **restart**
    $\mathbf{z}_2^\dagger \leftarrow (\lfloor \mathbf{u} \rceil_d - \lfloor \mathbf{u} - \mathbf{z}_2 \rceil_d) \bmod p$
    return $(\mathbf{z}_1, \mathbf{z}_2^\dagger, \mathbf{c})$
end function

BLISS: signature

function $\textsc{Sign}(\mu, pk = \mathbf{A},\ sk = \mathbf{S})$

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

$\mathbf{u} = \zeta \cdot \mathbf{a}_1 \cdot \mathbf{y}_1 + \mathbf{y}_2 \bmod 2q$ ▷ $\zeta = 1/(q-2)$

$\mathbf{c} \leftarrow H(\lfloor \mathbf{u} \rceil_d \bmod p, \mu)$ ▷ 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 $1/\big(M \exp(-\|\mathbf{Sc}\|/(2\sigma^2)) \cosh(\langle \mathbf{z}, \mathbf{Sc}\rangle/\sigma^2\big)$ otherwise **restart**

$\mathbf{z}_2^\dagger \leftarrow (\lfloor \mathbf{u} \rceil_d - \lfloor \mathbf{u} - \mathbf{z}_2 \rceil_d) \bmod p$

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

end function

BLISS: signature

function $\textsc{Sign}(\mu, pk = \mathbf{A}, sk = \mathbf{S})$
  $\mathbf{y}_1, \mathbf{y}_2 \leftarrow D^n_{\mathbb{Z},\sigma}$ ▷ Gaussian sampling
  $\mathbf{u} = \zeta \cdot \mathbf{a}_1 \cdot \mathbf{y}_1 + \mathbf{y}_2 \bmod 2q$ ▷ $\zeta = 1/(q-2)$
  $\mathbf{c} \leftarrow H(\lfloor \mathbf{u} \rceil_d \bmod p, \mu)$ ▷ 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
  $1/\left(M \exp(-\|\mathbf{Sc}\|/(2\sigma^2)) \cosh(\langle \mathbf{z}, \mathbf{Sc} \rangle / \sigma^2\right)$ otherwise **restart**
  $\mathbf{z}_2^\dagger \leftarrow (\lfloor \mathbf{u} \rceil_d - \lfloor \mathbf{u} - \mathbf{z}_2 \rceil_d) \bmod p$
  return $(\mathbf{z}_1, \mathbf{z}_2^\dagger, \mathbf{c})$
end function

BLISS: verification

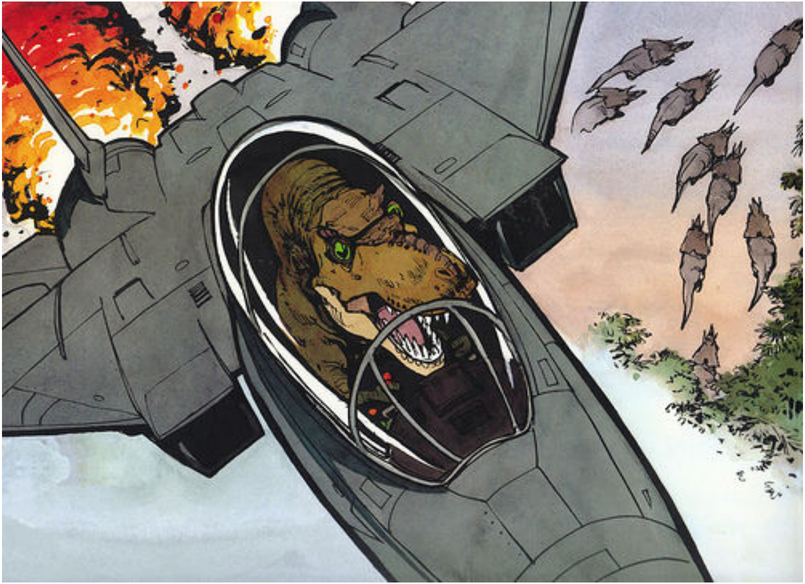
function $\textsc{Verify}(\mu, \mathbf{A}, (\mathbf{z}_1, \mathbf{z}_2^\dagger, \mathbf{c}))$
  if $\|(\mathbf{z}_1 | 2^d \cdot \mathbf{z}_2^\dagger)\|_2 > B_2$ **then** reject
  if $\|(\mathbf{z}_1 | 2^d \cdot \mathbf{z}_2^\dagger)\|_\infty > B_\infty$ **then** reject
  accept iff $\mathbf{c} = H(\lfloor \zeta \cdot \mathbf{a}_1 \cdot \mathbf{z}_1 + \zeta \cdot q \cdot \mathbf{c} \rceil_d + \mathbf{z}_2^\dagger \bmod p, \mu)$
end function

BLISS: parameters

- Parameters proposed by Ducas et al. for 128-bit security (BLISS–I & BLISS–II)
  - $n = 512$, $q = 12289$
  - $(\delta_1, \delta_2) = (0.3, 0)$ (density of $\mathbf{f}, \mathbf{g}$)
  - $\sigma = 215$ for BLISS–I, 107 for BLISS–II
  - $\kappa = 23$ (number of 1's in $\mathbf{c}$)

Outline

Let's break things!



Attacking $\mathbf{y}$

- $\mathbf{y}_1$ ($\equiv$discrete Gaussian) $\approx$ additive mask in

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

- Sampling: coefficient by coefficient
- Use fault injection to abort the sampling early $\Longrightarrow$ faulty signature with a low-degree $\mathbf{y}_1$
- Done by attacking:
  - Branching test of the loop (voltage spike, clock variation...)
  - Contents of the loop counter (lasers, x-rays...)

Attacking $\mathbf{y}$

- $\mathbf{y}_1$ ($\equiv$discrete Gaussian) $\approx$ additive mask in

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

- Sampling: coefficient by coefficient
- Use fault injection to abort the sampling early $\Longrightarrow$ faulty signature with a low-degree $\mathbf{y}_1$
- Done by attacking:
  - Branching test of the loop (voltage spike, clock variation...)
  - Contents of the loop counter (lasers, x-rays...)

Attacking $\mathbf{y}$

- $\mathbf{y}_1$ ($\equiv$discrete Gaussian) $\approx$ additive mask in

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

- Sampling: coefficient by coefficient
- Use fault injection to abort the sampling early $\Longrightarrow$ faulty signature with a low-degree $\mathbf{y}_1$
- Done by attacking:
  - Branching test of the loop (voltage spike, clock variation...)
  - Contents of the loop counter (lasers, x-rays...)

Attacking $\mathbf{y}$

- $\mathbf{y}_1$ ($\equiv$discrete Gaussian) $\approx$ additive mask in

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

- Sampling: coefficient by coefficient
- Use fault injection to abort the sampling early $\Longrightarrow$ faulty signature with a low-degree $\mathbf{y}_1$
- Done by attacking:
  - Branching test of the loop (voltage spike, clock variation...)
  - Contents of the loop counter (lasers, x-rays...)

Attack details (I)

- So let's say we get a signature generated with $\mathbf{y}_1$ of degree $m \ll n$
- If $\mathbf{c}$ is invertible (probability around $(1 - 1/q)^n \approx 96\%$), we can compute:

$$\mathbf{v} = \mathbf{c}^{-1}\mathbf{z}_1 \equiv \mathbf{c}^{-1}\mathbf{y}_1 + (-1)^b \mathbf{s}_1 \pmod{q}$$

- WLOG, $b = 0$ (equivalent keys)
- Since $\mathbf{s}_1$ is very short, $\mathbf{v}$ very close to the lattice L generated by $q\mathbb{Z}^n$ and $\mathbf{w}_i = \mathbf{c}^{-1}\mathbf{x}^i$, $i = 0, \ldots, m$

Attack details (I)

- So let's say we get a signature generated with $\mathbf{y}_1$ of degree $m \ll n$
- If $\mathbf{c}$ is invertible (probability around $(1-1/q)^n \approx 96\%$), we can compute:

$$\mathbf{v} = \mathbf{c}^{-1}\mathbf{z}_1 \equiv \mathbf{c}^{-1}\mathbf{y}_1 + (-1)^b \mathbf{s}_1 \pmod{q}$$

- WLOG, $b = 0$ (equivalent keys)
- Since $\mathbf{s}_1$ is very short, $\mathbf{v}$ very close to the lattice L generated by $q\mathbb{Z}^n$ and $\mathbf{w}_i = \mathbf{c}^{-1}\mathbf{x}^i$, $i = 0, \ldots, m$

Attack details (I)

- So let's say we get a signature generated with $\mathbf{y}_1$ of degree $m \ll n$
- If $\mathbf{c}$ is invertible (probability around $(1-1/q)^n \approx 96\%$), we can compute:

$$\mathbf{v} = \mathbf{c}^{-1}\mathbf{z}_1 \equiv \mathbf{c}^{-1}\mathbf{y}_1 + (-1)^b \mathbf{s}_1 \pmod{q}$$

- WLOG, $b = 0$ (equivalent keys)
- Since $\mathbf{s}_1$ is very short, $\mathbf{v}$ very close to the lattice L generated by $q\mathbb{Z}^n$ and $\mathbf{w}_i = \mathbf{c}^{-1}\mathbf{x}^i$, $i = 0, \ldots, m$

Attack details (I)

- So let's say we get a signature generated with $\mathbf{y}_1$ of degree $m \ll n$
- If $\mathbf{c}$ is invertible (probability around $(1-1/q)^n \approx 96\%$), we can compute:

$$\mathbf{v} = \mathbf{c}^{-1}\mathbf{z}_1 \equiv \mathbf{c}^{-1}\mathbf{y}_1 + (-1)^b\mathbf{s}_1 \pmod{q}$$

- WLOG, $b = 0$ (equivalent keys)
- Since $\mathbf{s}_1$ is very short, $\mathbf{v}$ very close to the lattice L generated by $q\mathbb{Z}^n$ and $\mathbf{w}_i = \mathbf{c}^{-1}\mathbf{x}^i$, $i = 0, \ldots, m$

Attack details (II)

L of dimension n: too large to apply lattice reduction

Attack details (II)

- Same relation on arbitrary subset of coefficients: we can reduce the dimension
- More precisely, fix a subset $I \subset \{0, \ldots, n-1\}$ of ℓ indices, and let $\varphi_I : \mathbb{Z}^n \to \mathbb{Z}^I$ be the obvious projection
- $\varphi_I(\mathbf{v})$ is close to the lattice generated by $\varphi_I(\mathbf{w}_i)$ and $q\mathbb{Z}^I$, and if ℓ is large enough, the difference should be $\varphi_I(\mathbf{s}_1)$.

Attack details (II)

- Same relation on arbitrary subset of coefficients: we can reduce the dimension
- More precisely, fix a subset $I \subset \{0, \ldots, n-1\}$ of ℓ indices, and let $\varphi_I : \mathbb{Z}^n \to \mathbb{Z}^I$ be the obvious projection
- $\varphi_I(\mathbf{v})$ is close to the lattice generated by $\varphi_I(\mathbf{w}_i)$ and $q\mathbb{Z}^I$, and if ℓ is large enough, the difference should be $\varphi_I(\mathbf{s}_1)$.

Attack details (II)

- Same relation on arbitrary subset of coefficients: we can reduce the dimension
- More precisely, fix a subset $I \subset \{0, \ldots, n-1\}$ of ℓ indices, and let $\varphi_I : \mathbb{Z}^n \to \mathbb{Z}^I$ be the obvious projection
- $\varphi_I(\mathbf{v})$ is close to the lattice generated by $\varphi_I(\mathbf{w}_i)$ and $q\mathbb{Z}^I$, and if ℓ is large enough, the difference should be $\varphi_I(\mathbf{s}_1)$.

Attack details (III)

- Solve this close vector problem using Babai nearest plane algorithm. Condition on ℓ to recover $\varphi_I(\mathbf{s}_1)$:

$$\ell + 1 \gtrsim \frac{m + 2 + \frac{\log\sqrt{\delta_1 + 4\delta_2}}{\log q}}{1 - \frac{\log\sqrt{2\pi e(\delta_1 + 4\delta_2)}}{\log q}}$$

Implementation results

- For BLISS–I and BLISS–II, this says $\ell \approx 1.09 \cdot m$
- In practice: works fine with LLL for $m \lesssim 60$ and with BKZ with $m \lesssim 100$
- Just apply the attack for several choices of I to recover all of $\mathbf{s}_1$, and subsequently $\mathbf{s}_2$: full key recovery with one faulty sig.!

Fault after iteration number $m =$	5	10	20	40	80	100
Theoretical minimum dimension $\ell_{\min}$	6	11	22	44	88	110
Dimension ℓ in our experiment	6	12	24	50	110	150
Lattice reduction algorithm	LLL	LLL	LLL	BKZ–20	BKZ–25	BKZ–25
Success probability (%)	99	100	100	100	100	98
Avg. CPU time to recover ℓ coeffs. (s)	0.005	0.022	0.23	7.3	941	33655
Avg. CPU time for full key recovery	0.5 s	1 s	5 s	80 s	80 min	38 h

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 on constrained devices, so some tricks have to be used

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 on constrained devices, so some tricks have to be used

Attack overview

- The optimized version of the rejection sampling used iterated Bernoulli trials on each of the bits of $\|\mathbf{Sc}\|^2$; as a result, we can read that value on an SPA trace

- This yields to the recovery of the relative norm $\mathbf{s} \cdot \bar{\mathbf{s}}$ of the secret key. Algorithmic number theoretic techniques (Howgrave-Graham–Szydlo) can then be used to retrive $\mathbf{s}$!

Attack overview

- The optimized version of the rejection sampling used iterated Bernoulli trials on each of the bits of $\|\mathbf{Sc}\|^2$; as a result, we can read that value on an SPA trace

- This yields to the recovery of the relative norm $\mathbf{s} \cdot \bar{\mathbf{s}}$ of the secret key. Algorithmic number theoretic techniques (Howgrave-Graham–Szydlo) can then be used to retrive $\mathbf{s}$!

BLISS rejection sampling

function 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 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)

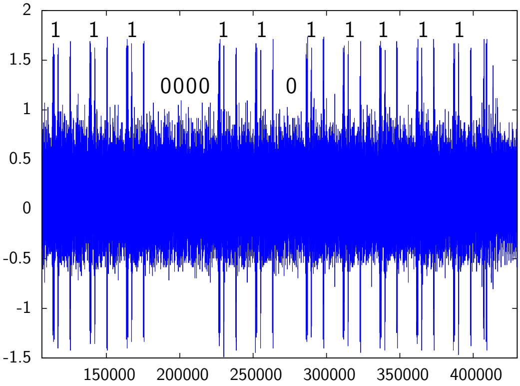
BLISS rejection sampling

function 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 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)

Experimental leakage



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

$$K - \|\mathbf{Sc}\|^2 = 46539 - 14404 = \overline{1110000110111 1}_2$$

Exploiting the leakage

- After collecting around 1024 traces, one obtains the value of $\mathbf{S} \cdot \overline{\mathbf{S}}$
- Algorithmic number theory: a generalized Howgrave-Graham–Szydlo algorithm allows to deduce $\mathbf{S}$ itself (up to a root of unity)
- Attack in polynomial time if the algebraic norm of $\mathbf{S}$ is easy to factor (e.g. semismooth: happens in a significant fraction of cases!)
- This is a full key recovery!

Exploiting the leakage

- After collecting around 1024 traces, one obtains the value of $\mathbf{S} \cdot \overline{\mathbf{S}}$
- Algorithmic number theory: a generalized Howgrave-Graham–Szydlo algorithm allows to deduce $\mathbf{S}$ itself (up to a root of unity)
- Attack in polynomial time if the algebraic norm of $\mathbf{S}$ is easy to factor (e.g. semismooth: happens in a significant fraction of cases!)
- This is a full key recovery!

Exploiting the leakage

- After collecting around 1024 traces, one obtains the value of $\mathbf{S} \cdot \overline{\mathbf{S}}$
- Algorithmic number theory: a generalized Howgrave-Graham–Szydlo algorithm allows to deduce **S** itself (up to a root of unity)
- Attack in polynomial time if the algebraic norm of **S** is easy to factor (e.g. semismooth: happens in a significant fraction of cases!)
- This is a full key recovery!

Exploiting the leakage

- After collecting around 1024 traces, one obtains the value of $\mathbf{S} \cdot \overline{\mathbf{S}}$
- Algorithmic number theory: a generalized Howgrave-Graham–Szydlo algorithm allows to deduce $\mathbf{S}$ itself (up to a root of unity)
- Attack in polynomial time if the algebraic norm of $\mathbf{S}$ is easy to factor (e.g. semismooth: happens in a significant fraction of cases!)
- This is a full key recovery!

Efficiency of the attack

Field size n	32	64	128	256	512
CPU time	0.6 s	13 s	21 min.	17h 22 min.	1.2 months (est.)
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

One could say that...

The security of BLISS implementation relies on the hardness of ...

Integer Factorization!

One could say that...

The security of BLISS implementation relies on the hardness of ...

Integer Factorization!

Outline

Introduction
- Implementation attacks
- Implementation attacks on lattice schemes

Physical attacks against BLISS
- The BLISS signature scheme
- Fault attack on the Gaussian sampling
- SCA on the rejection sampling

GPV-based schemes
- The signature scheme
- Fault attack on the Gaussian sampler

Conclusion and countermeasures

GPV-Based scheme

- Variant of Ducas-Lyubashevsky-Prest based on GPV-style lattice trapdoors.
- Defined once again over $\mathcal{R} = \mathbb{Z}[\mathbf{x}]/(\mathbf{x}^n + 1)$
- Secret key:

$$\mathbf{B} \leftarrow \begin{pmatrix} \mathbf{M_g} & -\mathbf{M_f} \\ \mathbf{M_G} & -\mathbf{M_F} \end{pmatrix} \in \mathbb{Z}^{2n \times 2n}$$

for $\mathbf{f} \leftarrow D^n_{\sigma_0}$, $\mathbf{g} \leftarrow D^n_{\sigma_0}$

$$f \cdot G - g \cdot F = q$$

Sign and Verify

function $\textsc{Sign}(\mu, sk = \mathbf{B})$
  $\mathbf{c} \leftarrow H(\mu) \in \mathbb{Z}_q^n$
  $(\mathbf{y}, \mathbf{z}) \leftarrow (\mathbf{c}, \mathbf{0}) - \textsc{GaussianSampler}(\mathbf{B}, \sigma, (\mathbf{c}, \mathbf{0}))$ ▷ $\mathbf{y}, \mathbf{z}$ *are short and satisfy* $\mathbf{y} + \mathbf{z} \cdot \mathbf{h} = \mathbf{c} \bmod q$
  return $\mathbf{z}$
end function

function $\textsc{Verify}(\mu, pk = \mathbf{h}, \mathbf{z})$
  accept iff $\|\mathbf{z}\|_2 + \|H(\mu) - \mathbf{z} \cdot \mathbf{h}\|_2 \leq \sigma\sqrt{2n}$
end function

Sign and Verify

function $\textsc{Sign}(\mu, sk = \mathbf{B})$
  $\mathbf{c} \leftarrow H(\mu) \in \mathbb{Z}_q^n$
  $(\mathbf{y}, \mathbf{z}) \leftarrow (\mathbf{c}, \mathbf{0}) - \textsc{GaussianSampler}(\mathbf{B}, \sigma, (\mathbf{c}, \mathbf{0}))$ ▷ $\mathbf{y}, \mathbf{z}$ *are short and satisfy* $\mathbf{y} + \mathbf{z} \cdot \mathbf{h} = \mathbf{c} \bmod q$
  return $\mathbf{z}$
end function

function $\textsc{Verify}(\mu, pk = \mathbf{h}, \mathbf{z})$
  accept iff $\|\mathbf{z}\|_2 + \|H(\mu) - \mathbf{z} \cdot \mathbf{h}\|_2 \leq \sigma\sqrt{2n}$
end function

Gaussian Sampling

function $\textsc{GaussianSampler}(\mathbf{B}, \sigma, \mathbf{c})$ $\triangleright$ $\mathbf{b}_i$ (resp. $\widetilde{\mathbf{b}}_i$) are the rows of $\mathbf{B}$ (resp. of its Gram–Schmidt matrix $\widetilde{\mathbf{B}}$)

  $\mathbf{v} \leftarrow \mathbf{0}$

  for $i = 2n$ **down to** 1 **do**

    $c' \leftarrow \langle \mathbf{c}, \mathbf{b_i} \rangle / \|\widetilde{\mathbf{b}}_i\|_2^2$

    $\sigma' \leftarrow \sigma / \|\widetilde{\mathbf{b}}_i\|_2$

    $r \leftarrow D_{\mathbb{Z},\sigma',c'}$

    $\mathbf{c} \leftarrow \mathbf{c} - r\mathbf{b}_i$ and $\mathbf{v} \leftarrow \mathbf{v} + r\mathbf{b}_i$

  end for

  return v $\triangleright$ $\mathbf{v}$ sampled according to the lattice Gaussian distribution $D_{\Lambda,\sigma,\mathbf{c}}$

end function

Gaussian Sampling

function $\textsc{GaussianSampler}(\mathbf{B}, \sigma, \mathbf{c})$ ▷ $\mathbf{b}_i$ (resp. $\widetilde{\mathbf{b}}_i$) are the rows of $\mathbf{B}$ (resp. of its Gram–Schmidt matrix $\widetilde{\mathbf{B}}$)
    $\mathbf{v} \leftarrow \mathbf{0}$
    for $i = 2n$ **down to** 1 **do**
        $c' \leftarrow \langle \mathbf{c}, \mathbf{b_i} \rangle / \|\widetilde{\mathbf{b}}_i\|_2^2$
        $\sigma' \leftarrow \sigma / \|\widetilde{\mathbf{b}}_i\|_2$
        $r \leftarrow D_{\mathbb{Z},\sigma',c'}$
        $\mathbf{c} \leftarrow \mathbf{c} - r\mathbf{b}_i$ and $\mathbf{v} \leftarrow \mathbf{v} + r\mathbf{b}_i$
    end for
        return v ▷ $\mathbf{v}$ sampled according to the lattice Gaussian distribution $D_{\Lambda,\sigma,\mathbf{c}}$
end function

Outline

Attacking the Gaussian sampler

- Correctly generated signature: element of the form

$$\mathbf{z} = \mathbf{R} \cdot \mathbf{f} + \mathbf{r} \cdot \mathbf{F} \in \mathbb{Z}[\mathbf{x}]/(\mathbf{x}^n + 1)$$

- Faults introduced after m iterations of the generation of r, R:

$$\mathbf{z} = r_0 \mathbf{x}^{n-1}\mathbf{F} + r_1 \mathbf{x}^{n-2}\mathbf{F} + \cdots + r_{m-1}\mathbf{x}^{n-m}\mathbf{F}.$$

- Belongs to lattice :

$$L = \mathrm{Span}(\mathbf{x}^{n-i}\mathbf{F})$$

for $1 \leq i \leq m$.

Attacking the Gaussian sampler

- Correctly generated signature: element of the form

$$\mathbf{z} = \mathbf{R} \cdot \mathbf{f} + \mathbf{r} \cdot \mathbf{F} \in \mathbb{Z}[\mathbf{x}]/(\mathbf{x}^n + 1)$$

- Faults introduced after m iterations of the generation of r, R:

$$\mathbf{z} = r_0 \mathbf{x}^{n-1} \mathbf{F} + r_1 \mathbf{x}^{n-2} \mathbf{F} + \cdots + r_{m-1} \mathbf{x}^{n-m} \mathbf{F}.$$

- Belongs to lattice :

$$L = \mathrm{Span}(\mathbf{x}^{n-i} \mathbf{F})$$

for $1 \leq i \leq m$.

Attacking the Gaussian sampler

- Correctly generated signature: element of the form

$$\mathbf{z} = \mathbf{R} \cdot \mathbf{f} + \mathbf{r} \cdot \mathbf{F} \in \mathbb{Z}[\mathbf{x}]/(\mathbf{x}^n + 1)$$

- Faults introduced after m iterations of the generation of r, R:

$$\mathbf{z} = r_0 \mathbf{x}^{n-1} \mathbf{F} + r_1 \mathbf{x}^{n-2} \mathbf{F} + \cdots + r_{m-1} \mathbf{x}^{n-m} \mathbf{F}.$$

- Belongs to lattice :

$$L = \mathrm{Span}(\mathbf{x}^{n-i} \mathbf{F})$$

for $1 \leq i \leq m$.

Multiple faulted signatures?

- $\mathbf{z}^{(1)}, \ldots, \mathbf{z}^{(\ell)}$ faulty signatures.
- With probability $\geq \prod_{k=l-m+1}^{+\infty} \frac{1}{\zeta(k)}$ generates L. [*Maze, Rosenthal, Wagner*]
- SVP of L should be one of the $\mathbf{x}^{n-i}\mathbf{F}$ for $1 \leq i \leq m$.

$\Longrightarrow$ Full recovery of a basis $(\zeta f, \zeta g, \zeta F, \zeta G)$ for a $\zeta = \pm \mathbf{x}^{\alpha}$.
(*equivalent keys*)

Multiple faulted signatures?

- $\mathbf{z}^{(1)}, \ldots, \mathbf{z}^{(\ell)}$ faulty signatures.
- With probability $\geq \prod_{k=l-m+1}^{+\infty} \frac{1}{\zeta(k)}$ generates L. [*Maze, Rosenthal, Wagner*]
- SVP of L should be one of the $\mathbf{x}^{n-i}\mathbf{F}$ for $1 \leq i \leq m$.

$\Longrightarrow$ Full recovery of a basis $(\zeta f, \zeta g, \zeta F, \zeta G)$ for a $\zeta = \pm \mathbf{x}^{\alpha}$.
(*equivalent keys*)

Multiple faulted signatures?

- $\mathbf{z}^{(1)}, \ldots, \mathbf{z}^{(\ell)}$ faulty signatures.
- With probability $\geq \prod_{k=l-m+1}^{+\infty} \frac{1}{\zeta(k)}$ generates L. [*Maze, Rosenthal, Wagner*]
- SVP of L should be one of the $\mathbf{x}^{n-i}\mathbf{F}$ for $1 \leq i \leq m$.

$\Longrightarrow$ Full recovery of a basis $(\zeta f, \zeta g, \zeta F, \zeta G)$ for a $\zeta = \pm \mathbf{x}^{\alpha}$.
(*equivalent keys*)

Multiple faulted signatures?

- $\mathbf{z}^{(1)}, \ldots, \mathbf{z}^{(\ell)}$ faulty signatures.
- With probability $\geq \prod_{k=l-m+1}^{+\infty} \frac{1}{\zeta(k)}$ generates L. [*Maze, Rosenthal, Wagner*]
- SVP of L should be one of the $\mathbf{x}^{n-i}\mathbf{F}$ for $1 \leq i \leq m$.

$\Longrightarrow$ Full recovery of a basis $(\zeta f, \zeta g, \zeta F, \zeta G)$ for a $\zeta = \pm \mathbf{x}^{\alpha}$.
(*equivalent keys*)

In practice

Fault after iteration number $m =$ Lattice reduction algorithm	2 LLL	5 LLL	10 LLL	20 LLL	40 LLL	60 LLL	80 BKZ–20	100 BKZ–20
Success probability for $\ell = m + 1$ (%)	75	77	90	93	94	94	95	95
Avg. CPU time for $\ell = m + 1$ (s)	0.001	0.003	0.016	0.19	2.1	8.1	21.7	104
Success probability for $\ell = m + 2$ (%)	89	95	100	100	99	99	100	100
Avg. CPU time for $\ell = m + 2$ (s)	0.001	0.003	0.017	0.19	2.1	8.2	21.6	146

Conclusion and countermeasures

- We described faults and SCA against BLISS signatures
- Possible countermeasures?
- Against Faults:
  - check that the result has $> (1-\varepsilon)\cdot n$ non zero coeffs.
  - randomize the order of generation of the coefficients? (still risky)
  - use double loop counters!
- Against SCA:
  - compute rejection probability with floating point arithmetic (slow)
  - use a constant-time Bernoulli sampling (doable)
  - prefer a scheme with simpler structure (GLP) and use masking

Conclusion and countermeasures

- We described faults and SCA against BLISS signatures
- Possible countermeasures?
- Against Faults:
  - check that the result has $> (1 - \varepsilon) \cdot n$ non zero coeffs.
  - randomize the order of generation of the coefficients? (still risky)
  - use double loop counters!
- Against SCA:
  - compute rejection probability with floating point arithmetic (slow)
  - use a constant-time Bernoulli sampling (doable)
  - prefer a scheme with simpler structure (GLP) and use masking

Conclusion and countermeasures

- We described faults and SCA against BLISS signatures
- Possible countermeasures?
- Against Faults:
  - check that the result has $> (1 - \varepsilon) \cdot n$ non zero coeffs.
  - randomize the order of generation of the coefficients? (still risky)
  - use double loop counters!
- Against SCA:
  - compute rejection probability with floating point arithmetic (slow)
  - use a constant-time Bernoulli sampling (doable)
  - prefer a scheme with simpler structure (GLP) and use masking

Conclusion and countermeasures

- Important to investigate implementation attacks on lattice schemes
- Physical attack resistance should be part of the design goals for practical schemes

Conclusion and countermeasures

- Important to investigate implementation attacks on lattice schemes
- Physical attack resistance should be part of the design goals for practical schemes

Conclusion and countermeasures

- Thank you !