Improving Side-channel Attacks on Lattice-based Cryptography

Leon Groot Bruinderink
based on joint work with Peter Pessl and Yuval Yarom

June 2nd, 2017

Public-key Cryptography

- Important part of secure internet
- Encryption: keep information confidential
- Digital signatures:
    - Authentication (prove authorship of message)
    - Integrity (prove data not changed)
- For example, Bitcoin relies heavily on digital signatures

Lattice-based Cryptography

- Quantum computers pose threat to current cryptography on the internet (DH, RSA, ECC)
- Lattice-based cryptography: promising post-quantum secure alternative.
- Active research on theoretical and practical security.
- Security of implementations largely unexplored.

Side-channel attacks

- Use physical information leakage from implementations
- Timing attacks can be done remotely
- Use that to perform a key-recovery

This talk

- Show side-channel attack on lattice-based signature scheme BLISS
- Model side-channel simply as additional knowledge to attacker
- Show the used key-recovery techniques

BLISS

BLISS Lattice-based Signature Scheme

- Bimodal Lattice Signature Scheme (BLISS)
- Implementations available via NTRU lattices (polynomials in $R_q = \mathbb{Z}_q[x]/(x^n + 1)$, $n = 2^r$, prime q).
- For $f, g \in R_q = \mathbb{Z}_q[x]/(x^n + 1)$:

$$f \cdot g = \mathbf{f}\mathrm{G} = \mathbf{g}\mathrm{F}$$

where $\mathrm{F}, \mathrm{G} \in \mathbb{Z}_q^{n \times n}$, whose columns are rotations of $\mathbf{f}, \mathbf{g}$, with possibly opposite sign:

$$\mathrm{F} = \begin{bmatrix} f_0 & -f_{n-1} & \dots & -f_1 \\ f_1 & f_0 & \dots & -f_2 \\ \dots & \dots & \dots & \dots \\ f_{n-1} & f_{n-2} & \dots & f_0 \end{bmatrix}$$

BLISS Lattice-based Signature Scheme

- Secret key $\mathbf{S} = (f, 2g+1) \in R_q^2$ with f, g sparse and typically entries in $\{\pm 1, 0\}$
- Public key $\mathbf{A} = (a_1, a_2) \in R_q^2$ satisfying:

$$a_1 s_1 + a_2 s_2 \equiv q \bmod 2q$$

- Computed as $a_q = (2g+1)/f \bmod q$ (restart if f not invertible) and $\mathbf{A} = (2a_q, q-2)$.
- Attacker can validate correctness for candidate of key f with the public key and compute $2g+1$.
- Both $-\mathbf{S}$ and $\mathbf{S}$ are valid as secret key.

BLISS Lattice-based Signature Scheme

- Simplified version of the BLISS signature algorithm for message μ:

BLISS Lattice-based Signature Scheme

- Simplified version of the BLISS signature algorithm for message μ:

1. Sample $\mathbf{y} \leftarrow D_{\mathbb{Z}^n,\sigma}$.

BLISS Lattice-based Signature Scheme

- Simplified version of the BLISS signature algorithm for message μ:

1. Sample $\mathbf{y} \leftarrow D_{\mathbb{Z}^n,\sigma}$.
2. Construct vector $\mathbf{u}$, using $\mathbf{y}$ and public key $\mathbf{A}$.

BLISS Lattice-based Signature Scheme

- Simplified version of the BLISS signature algorithm for message μ:

1. Sample $\mathbf{y} \leftarrow D_{\mathbb{Z}^n,\sigma}$.
2. Construct vector $\mathbf{u}$, using $\mathbf{y}$ and public key $\mathbf{A}$.
3. Construct challenge $\mathbf{c} = H(\mathbf{u}, \mu) \in \{0,1\}^n$ with $||\mathbf{c}||_1 = \kappa$

- Simplified version of the BLISS signature algorithm for message μ:

1. Sample $\mathbf{y} \leftarrow D_{\mathbb{Z}^n,\sigma}$.
2. Construct vector $\mathbf{u}$, using $\mathbf{y}$ and public key $\mathbf{A}$.
3. Construct challenge $\mathbf{c} = H(\mathbf{u}, \mu) \in \{0,1\}^n$ with $||\mathbf{c}||_1 = \kappa$
4. Generate a random bit b. Set $\mathbf{z} = \mathbf{y} + (-1)^b \mathbf{s}_1 \cdot \mathbf{c}$

BLISS Lattice-based Signature Scheme

- Simplified version of the BLISS signature algorithm for message μ:

1. Sample $\mathbf{y} \leftarrow D_{\mathbb{Z}^n,\sigma}$.
2. Construct vector $\mathbf{u}$, using $\mathbf{y}$ and public key $\mathbf{A}$.
3. Construct challenge $\mathbf{c} = H(\mathbf{u}, \mu) \in \{0,1\}^n$ with $||\mathbf{c}||_1 = \kappa$
4. Generate a random bit b. Set $\mathbf{z} = \mathbf{y} + (-1)^b \mathbf{s}_1 \cdot \mathbf{c}$
5. Return signature $(\mathbf{z}, \mathbf{c})$ for μ.

BLISS Lattice-based Signature Scheme

- Simplified version of the BLISS signature algorithm for message μ:

1. Sample $\mathbf{y} \leftarrow D_{\mathbb{Z}^n,\sigma}$.
2. Construct vector $\mathbf{u}$, using $\mathbf{y}$ and public key $\mathbf{A}$.
3. Construct challenge $\mathbf{c} = H(\mathbf{u}, \mu) \in \{0,1\}^n$ with $||\mathbf{c}||_1 = \kappa$
4. Generate a random bit b. Set $\mathbf{z} = \mathbf{y} + (-1)^b \mathbf{s}_1 \cdot \mathbf{c}$
5. Return signature $(\mathbf{z}, \mathbf{c})$ for μ.

- $\mathbf{s}_1 \cdot \mathbf{c} = \mathbf{s}_1 \mathrm{C}$ over $\mathbb{Z}$ for matrix $\mathrm{C} \in \{-1, 0, 1\}^{n \times n}$.

BLISS Lattice-based Signature Scheme

- Simplified version of the BLISS signature algorithm for message μ:

1. Sample $\mathbf{y} \leftarrow D_{\mathbb{Z}^n,\sigma}$.
2. Construct vector $\mathbf{u}$, using $\mathbf{y}$ and public key $\mathbf{A}$.
3. Construct challenge $\mathbf{c} = H(\mathbf{u}, \mu) \in \{0,1\}^n$ with $||\mathbf{c}||_1 = \kappa$
4. Generate a random bit b. Set $\mathbf{z} = \mathbf{y} + (-1)^b \mathbf{s}_1 \cdot \mathbf{c}$
5. Return signature $(\mathbf{z}, \mathbf{c})$ for μ.

- $\mathbf{s}_1 \cdot \mathbf{c} = \mathbf{s}_1 \mathrm{C}$ over $\mathbb{Z}$ for matrix $\mathrm{C} \in \{-1, 0, 1\}^{n \times n}$.
- Equation in signature over $\mathbb{Z}$:

$$\mathbf{z} = \mathbf{y} + (-1)^b \mathbf{s}_1 \mathrm{C}$$

where the unknowns for the attacker are $\mathbf{y}, b, \mathbf{s}_1$

Discrete Gaussian Distribution

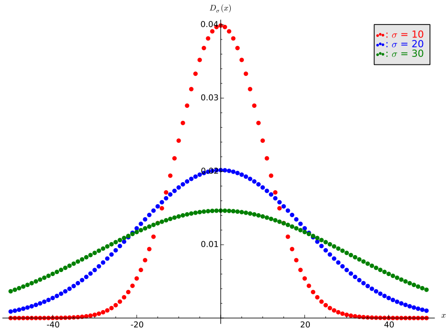


Figure 1: Discrete Gaussian distribution

- Step 1 in signature algorithm: $\mathbf{y} \leftarrow D_{\mathbb{Z}^m,\sigma}$
- This is required to achieve (provable) security and small signature size.
- Not straightforward to do in practice: high precision required.
- But how do we use additional knowledge of $\mathbf{y}$ to find $\mathbf{s}$?

Attack Scenario's

Attack Scenario 1

- Signature equation: $\mathbf{z} = \mathbf{y} + (-1)^b \mathbf{s}\mathrm{C}$

Scenario 1:
We can determine **y** completely from a side-channel attack

Attack Scenario 1

- Signature equation: $\mathbf{z} = \mathbf{y} + (-1)^b \mathbf{s}\mathrm{C}$

Scenario 1:
We can determine $\mathbf{y}$ completely from a side-channel attack

- Only need one signature.
- Solve equation $(-1)^b(\mathbf{z} - \mathbf{y}) = \mathbf{s}\mathrm{C}$ for $\mathbf{s}$.
- But unlikely...(?)

Attack Scenario 2

- System of n equations over $\mathbb{Z}$:

$$\underbrace{\begin{bmatrix} z_0 \\ z_1 \\ \dots \\ z_{n-1} \end{bmatrix}}_{\text{Signature}} = \underbrace{\begin{bmatrix} y_0 \\ y_1 \\ \dots \\ y_{n-1} \end{bmatrix}}_{\text{Noise}} + \underbrace{(-1)^b}_{\text{Sign}} \underbrace{\begin{bmatrix} - & \mathbf{c}_0 & - \\ - & \mathbf{c}_1 & - \\ - & \dots & - \\ - & \mathbf{c}_{n-1} & - \end{bmatrix}}_{\text{Challenge}} \cdot \underbrace{\begin{bmatrix} s_0 \\ s_1 \\ \dots \\ s_{n-1} \end{bmatrix}}_{\text{Secret}}$$

Scenario 2:
For some coefficients an attacker can determine y_i.

Attack Scenario 2

- System of n equations over $\mathbb{Z}$:

$$\underbrace{\begin{bmatrix} z_0 \\ z_1 \\ \dots \\ z_{n-1} \end{bmatrix}}_{\text{Signature}} = \underbrace{\begin{bmatrix} y_0 \\ y_1 \\ \dots \\ y_{n-1} \end{bmatrix}}_{\text{Noise}} + \underbrace{(-1)^b}_{\text{Sign}} \underbrace{\begin{bmatrix} - & \mathbf{c}_0 & - \\ - & \mathbf{c}_1 & - \\ - & \dots & - \\ - & \mathbf{c}_{n-1} & - \end{bmatrix}}_{\text{Challenge}} \cdot \underbrace{\begin{bmatrix} s_0 \\ s_1 \\ \dots \\ s_{n-1} \end{bmatrix}}_{\text{Secret}}$$

Scenario 2:
For some coefficients an attacker can determine y_i.

- Zoom in on coordinate-wise equalities:

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

- If we know y_i, we save $\zeta_k = \mathbf{c}_i$ in a list with y_i and z_i.

Attack Scenario 2

- We can acquire enough of these vectors from multiple signatures and form:

$$\begin{bmatrix} (-1)^{b_0}(z_0 - y_0) \\ (-1)^{b_1}(z_1 - y_1) \\ \dots \\ (-1)^{b_{n-1}}(z_{n-1} - y_{n-1}) \end{bmatrix} = \begin{bmatrix} - & \zeta_0 & - \\ - & \zeta_1 & - \\ - & \dots & - \\ - & \zeta_{n-1} & - \end{bmatrix} \cdot \begin{bmatrix} s_0 \\ s_1 \\ \dots \\ s_{n-1} \end{bmatrix}$$

- Unfortunately: all bits b_i are unknown.

Attack Scenario 2

- We can acquire enough of these vectors from multiple signatures and form:

$$\begin{bmatrix} (-1)^{b_0}(z_0 - y_0) \\ (-1)^{b_1}(z_1 - y_1) \\ \dots \\ (-1)^{b_{n-1}}(z_{n-1} - y_{n-1}) \end{bmatrix} = \begin{bmatrix} - & \zeta_0 & - \\ - & \zeta_1 & - \\ - & \dots & - \\ - & \zeta_{n-1} & - \end{bmatrix} \cdot \begin{bmatrix} s_0 \\ s_1 \\ \dots \\ s_{n-1} \end{bmatrix}$$

- Unfortunately: all bits b_i are unknown.
- Trick: if we know y_i, we can *be selective* and ensure that $z_i = y_i$, before saving $\zeta_k = \mathbf{c}_i$ in our list.
- We can eliminate b:

$$(-1)^b(z_i - y_i) = 0 = \langle \zeta_k, \mathbf{s} \rangle$$

Attack Scenario 2

- If we know y_i and $z_i = y_i$: we save $\zeta_k = \mathbf{c}_i$.
- Acquire enough of these vectors from multiple signatures and we have equation:

$$\mathbf{s}\mathrm{L} = \mathbf{0}$$

- With very high probability: secret vector $\mathbf{s}$ is the only vector in the integer (left) kernel of L.

Attack Scenario 3

- Signature equation over $\mathbb{Z}$: $\mathbf{z} = \mathbf{y} + (-1)^b \mathrm{C}\mathbf{s}$.
- Let us go one step further:

Scenario 3:
For some coefficients an attacker knows $y_i \in \{\gamma, \gamma + 1\}$
and with high probability, $y_i = \gamma$

Attack Scenario 3

- Apply same method as previous:
- If we know $y_i \in \{\gamma, \gamma + 1\}$ and $z_i = \gamma$: we save $\zeta_k = \mathbf{c}_i$.
- Now $\mathbf{s}\mathrm{L}$ is not an all-zero vector, but it is small.
- Use LLL-algorithm to compute small vectors, search for **s** in the unitary transformation matrix.
- Verify correctness with public key.

Results of Attacking BLISS

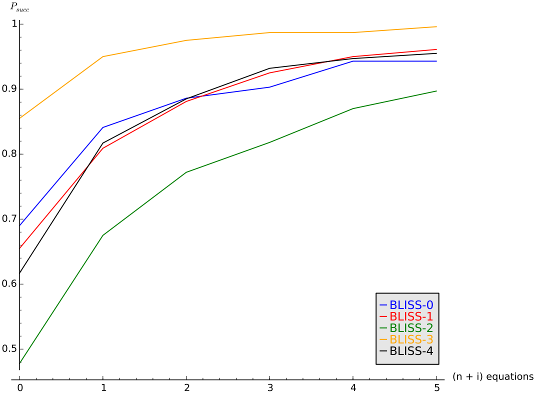


Figure 2: Success probability of LLL

- Previous attack scenario 2 and 3 were achievable in real-life

Improving Side-channel Attacks on BLISS

BLISS-B: accelerating signatures

- A new variant BLISS-B proposed, accelerating signing time by 2.8.
- Recall signature $(\mathbf{z}, \mathbf{c})$ with:

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

where the unknowns for the attacker are $\mathbf{y}, b, \mathbf{s}_1$

BLISS-B: accelerating signatures

- A new variant BLISS-B proposed, accelerating signing time by 2.8.
- Recall signature $(\mathbf{z}, \mathbf{c})$ with:

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

  where the unknowns for the attacker are $\mathbf{y}, b, \mathbf{s}_1$
- In BLISS-B, this is transformed to signature $(\mathbf{z}, \mathbf{c})$ with:

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

  where $\mathbf{c} \equiv \mathbf{c}^\star \bmod 2$.
- Unknowns to attacker now are $\mathbf{y}, b, \mathbf{s}_1$ **and** the signs of $\mathbf{c}^\star$.

BLISS-B: accelerating signatures

- A new variant BLISS-B proposed, accelerating signing time by 2.8.
- Recall signature $(\mathbf{z}, \mathbf{c})$ with:

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

  where the unknowns for the attacker are $\mathbf{y}, b, \mathbf{s}_1$
- In BLISS-B, this is transformed to signature $(\mathbf{z}, \mathbf{c})$ with:

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

  where $\mathbf{c} \equiv \mathbf{c}^\star \bmod 2$.
- Unknowns to attacker now are $\mathbf{y}, b, \mathbf{s}_1$ **and** the signs of $\mathbf{c}^\star$.
- The attacker cannot build the matrix/lattice basis in previous attacks!

A new key-recovery attack on BLISS-B: step 1

- Assume (possibly erroneous) information on y_i
- Coordinate-wise equalities in signature:

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

where attacker knows $\mathbf{c}_i \equiv \mathbf{c}_i^\star \bmod 2$ and z_i

A new key-recovery attack on BLISS-B: step 1

- Assume (possibly erroneous) information on y_i
- Coordinate-wise equalities in signature:

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

  where attacker knows $\mathbf{c}_i \equiv \mathbf{c}_i^\star \bmod 2$ and z_i
- Step 1: perform previous attack over $GF(2)$!
- No need of requiring $z_i = y_i$ (with high probability)
- Instead of LLL, use a LPN-solver
- This part gives the secret $\tilde{\mathbf{s}} \equiv \mathbf{s} \bmod 2$

A new key-recovery attack on BLISS-B: step 2

- Assume (possibly erroneous) information on y_i
- Coordinate-wise equalities in signature:

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

where attacker knows $\mathbf{c}_i \equiv \mathbf{c}_i^\star \bmod 2$ and z_i

- Attacker also knows $\tilde{\mathbf{s}} \equiv \mathbf{s} \bmod 2$ from step 1
- But keys can have $\mathbf{s}_i = \pm 2$, which are not detected by step 1.

A new key-recovery attack on BLISS-B: step 2

- Assume (possibly erroneous) information on y_i
- Coordinate-wise equalities in signature:

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

  where attacker knows $\mathbf{c}_i \equiv \mathbf{c}_i^\star \bmod 2$ and z_i
- Attacker also knows $\tilde{\mathbf{s}} \equiv \mathbf{s} \bmod 2$ from step 1
- But keys can have $\mathbf{s}_i = \pm 2$, which are not detected by step 1.
- Suppose

$$|z_i - y_i| = |\langle \mathbf{c}_i^\star, \mathbf{s} \rangle| > |\langle \mathbf{c}_i, \tilde{\mathbf{s}} \rangle|$$

A new key-recovery attack on BLISS-B: step 2

- Assume (possibly erroneous) information on y_i
- Coordinate-wise equalities in signature:

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

where attacker knows $\mathbf{c}_i \equiv \mathbf{c}_i^\star \bmod 2$ and z_i

- Attacker also knows $\tilde{\mathbf{s}} \equiv \mathbf{s} \bmod 2$ from step 1
- But keys can have $\mathbf{s}_i = \pm 2$, which are not detected by step 1.
- Suppose

$$|z_i - y_i| = |\langle \mathbf{c}_i^\star, \mathbf{s} \rangle| > |\langle \mathbf{c}_i, \tilde{\mathbf{s}} \rangle|$$

- There has to be *at least* one factor ± 2 making up for difference!
- Save $\mathbf{c}_i$ in a list

A new key-recovery attack on BLISS-B: step 2

- Suppose
$$|z_i - y_i| = |\langle \mathbf{c}_i^\star, \mathbf{s}\rangle| > |\langle \mathbf{c}_i, \tilde{\mathbf{s}}\rangle|$$
- Save $\mathbf{c}_i$ in a list
- Acquire many of these events...
- Two ways of extracting all coordinates of $\mathbf{s}$ that are ± 2:
  - Integer Programming solver
  - Statistical approach
- This part of the attack gives all magnitudes of the secret: $|\mathbf{s}|$

A new key-recovery attack on BLISS-B: final step

- So far, the attacker knows all magnitudes of the secret: $|\mathbf{s}| = |\mathbf{s}_1|$
- For the final step, we use the public key $\mathbf{A} = (2a_q, q-2)$ and $|\mathbf{s}_1|$.

A new key-recovery attack on BLISS-B: final step

- So far, the attacker knows all magnitudes of the secret: $|\mathbf{s}| = |\mathbf{s}_1|$
- For the final step, we use the public key $\mathbf{A} = (2a_q, q-2)$ and $|\mathbf{s}_1|$.
- Recall the key-relation over $R_q = \mathbb{Z}_q[x]/(x^n+1)$

$$a_q = s_2/s_1 = (2g+1)/f \bmod q$$

- Can model this relation as $\mathbf{A}_1\mathbf{s}_1 = \mathbf{s}_2$ for matrix $\mathbf{A}_1$ (q-ary lattice)

A new key-recovery attack on BLISS-B: final step

- So far, the attacker knows all magnitudes of the secret: $|\mathbf{s}| = |\mathbf{s}_1|$
- For the final step, we use the public key $\mathbf{A} = (2a_q, q-2)$ and $|\mathbf{s}_1|$.
- Recall the key-relation over $R_q = \mathbb{Z}_q[x]/(x^n+1)$

$$a_q = s_2/s_1 = (2g+1)/f \bmod q$$

- Can model this relation as $\mathbf{A}_1 \mathbf{s}_1 = \mathbf{s}_2$ for matrix $\mathbf{A}_1$ (q-ary lattice)
- Final step a: remove all columns j of $\mathbf{A}_1$ where $|\mathbf{s}_1|_j = 0$

A new key-recovery attack on BLISS-B: final step

- So far, the attacker knows all magnitudes of the secret: $|\mathbf{s}| = |\mathbf{s}_1|$
- For the final step, we use the public key $\mathbf{A} = (2a_q, q-2)$ and $|\mathbf{s}_1|$.
- Recall the key-relation over $R_q = \mathbb{Z}_q[x]/(x^n+1)$

$$a_q = s_2/s_1 = (2g+1)/f \bmod q$$

- Can model this relation as $\mathbf{A}_1\mathbf{s}_1 = \mathbf{s}_2$ for matrix $\mathbf{A}_1$ (q-ary lattice)
- Final step a: remove all columns j of $\mathbf{A}_1$ where $|\mathbf{s}_1|_j = 0$
- Final step b: remove some rows of $\mathbf{A}_1$ and try to solve $\mathbf{A}_1^*\mathbf{s}_1^* = \mathbf{s}_2^*$ using BKZ. This gives $\mathbf{s}_1^*$ which completely recovers $\mathbf{s}_1$

A new key-recovery attack on BLISS-B: final step

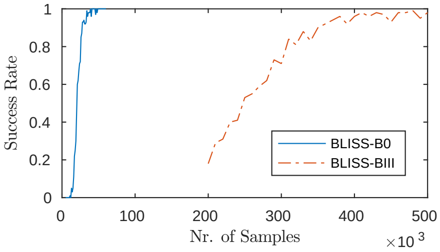
- So far, the attacker knows all magnitudes of the secret: $|\mathbf{s}| = |\mathbf{s}_1|$
- For the final step, we use the public key $\mathbf{A} = (2a_q, q-2)$ and $|\mathbf{s}_1|$.
- Recall the key-relation over $R_q = \mathbb{Z}_q[x]/(x^n+1)$

$$a_q = s_2/s_1 = (2g+1)/f \bmod q$$

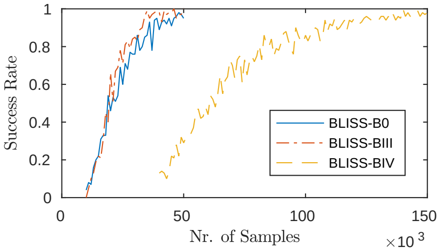
- Can model this relation as $\mathbf{A}_1\mathbf{s}_1 = \mathbf{s}_2$ for matrix $\mathbf{A}_1$ (q-ary lattice)
- Final step a: remove all columns j of $\mathbf{A}_1$ where $|\mathbf{s}_1|_j = 0$
- Final step b: remove some rows of $\mathbf{A}_1$ and try to solve $\mathbf{A}_1^*\mathbf{s}_1^* = \mathbf{s}_2^*$ using BKZ. This gives $\mathbf{s}_1^*$ which completely recovers $\mathbf{s}_1$
- Final step c: compute $\mathbf{s}_2$ by $a_q \cdot s_1$

Results of the full attack on BLISS-B

- BLISS-B implemented in strongSwan (library for secure VPN)
- Performed full real-life attack using previous steps on strongSwan



(a) Linear Programming



(b) Statistical

Figure 3: Success rate for recovery of ± 2 coefficients

Questions?