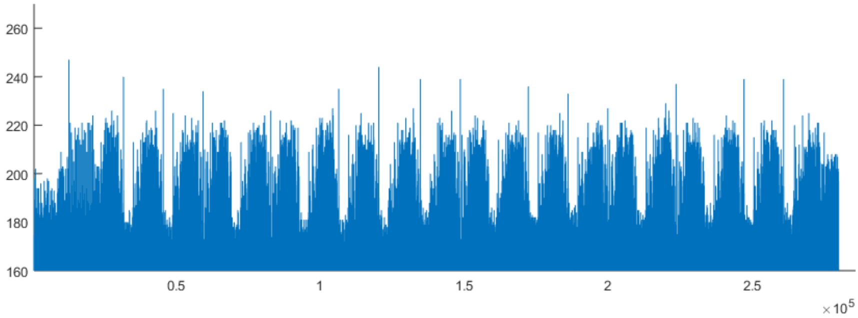
SCIENCE
PASSION
TECHNOLOGY

More Practical Single-Trace Attacks on the Number Theoretic Transform

Peter Pessl, Robert Primas
Graz University of Technology

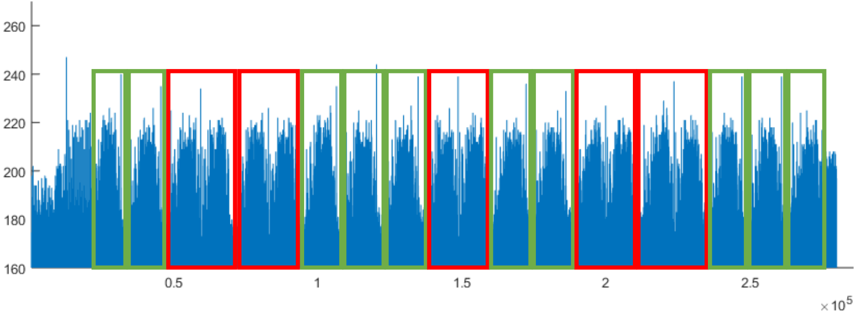
LATINCRYPT 2019, October 02

Public-Key Crypto and Side-Channel Attacks



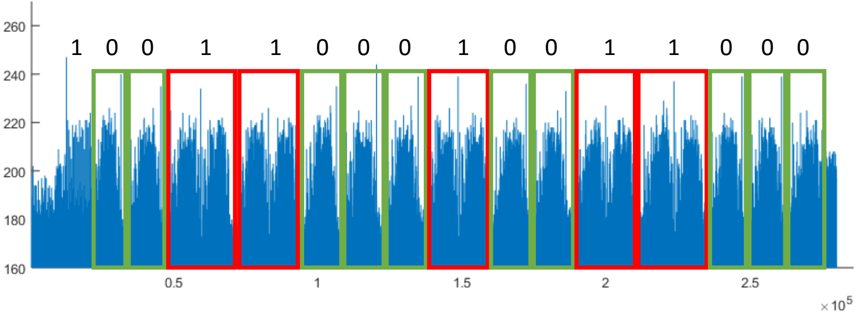
Power consumption trace of RSA decryption

Public-Key Crypto and Side-Channel Attacks



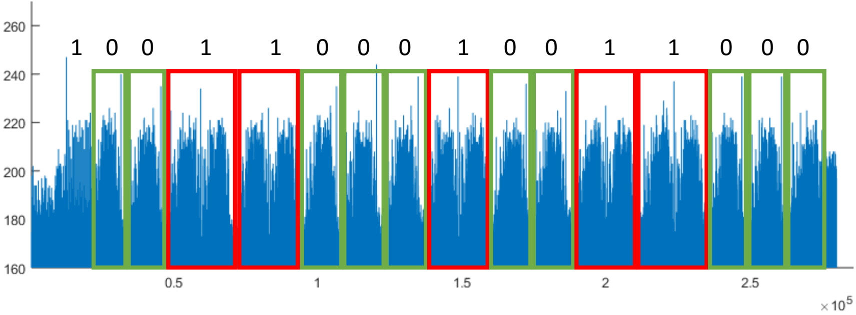
Power consumption trace of RSA decryption

Public-Key Crypto and Side-Channel Attacks



Power consumption trace of RSA decryption

Public-Key Crypto and Side-Channel Attacks



Power consumption trace of RSA decryption

Single-trace attacks are still a prime threat!

But RSA is old news anyway. . .

- Lattice-based cryptography
  - promising post-quantum replacement
  - implementations: fast and *constant time / control flow*
- Do we still need to worry about single-trace attacks?
  - no more instruction leakage
  - protection efforts towards differential (multi-trace) attacks

But RSA is old news anyway. . .

- Lattice-based cryptography
  - promising post-quantum replacement
  - implementations: fast and *constant time / control flow*
- Do we still need to worry about single-trace attacks?
  - no more instruction leakage
  - protection efforts towards differential (multi-trace) attacks

Previously: yes, but

- Our previous work: single-trace attack on the NTT
  - **N**umber **T**heoretic **T**ransform, common in many lattice schemes
  - combine *template attacks* (device profiling) with *belief propagation*
- but. . .
  - attacked variable-time implementation
  - large templating effort ($\approx$ a million multivariate templates)

Previously: yes, but

- Our previous work: single-trace attack on the NTT
  - **N**umber **T**heoretic **T**ransform, common in many lattice schemes
  - combine *template attacks* (device profiling) with *belief propagation*
- but. . .
  - attacked variable-time implementation
  - large templating effort ($\approx$ a million multivariate templates)

Previously: yes, but

- Our previous work: single-trace attack on the NTT
  - **N**umber **T**heoretic **T**ransform, common in many lattice schemes
  - combine *template attacks* (device profiling) with *belief propagation*
- but...
  - attacked variable-time implementation
  - large templating effort ($\approx$ a million multivariate templates)

Can we do better?

Our Contribution

- Improve upon previous attack
  - several improvements to belief propagation in this context
  - change targets: encryption instead of decryption
- Attack constant-time ASM-optimized Kyber implementation
  - massively reduced templating effort

Our Contribution

- Improve upon previous attack
  - several improvements to belief propagation in this context
  - change targets: encryption instead of decryption
- Attack constant-time ASM-optimized Kyber implementation
  - massively reduced templating effort

Lattice-based Encryption (LPR, NewHope, Kyber, . . .)

"Noisy ElGamal" with polynomials in $\mathbb{Z}_q[x]/\langle x^n + 1\rangle$

Key Generation: generate small *error polynomials* s, e

$t = a \cdot s + e$

$\text{pk} = (a, t), \ \text{sk} = s$

Encryption: generate small *error polynomials* r, e_1, e_2

$c_1 = a \cdot r + e_1$

$c_2 = t \cdot r + e_2 + m$

Decryption: $m \approx c_2 - s \cdot c_1$

Lattice-based Encryption (LPR, NewHope, Kyber, . . .)

"Noisy ElGamal" with polynomials in $\mathbb{Z}_q[x]/\langle x^n + 1\rangle$

Key Generation: generate small *error polynomials* s, e

$t = a \cdot s + e$

$\mathrm{pk} = (a, t),\ \mathrm{sk} = s$

Encryption: generate small *error polynomials* r, e_1, e_2

$c_1 = a \cdot r + e_1$

$c_2 = t \cdot r + e_2 + m$

Decryption: $m \approx c_2 - s \cdot c_1$

Lattice-based Encryption (LPR, NewHope, Kyber, . . .)

"Noisy ElGamal" with polynomials in $\mathbb{Z}_q[x]/\langle x^n + 1\rangle$

Key Generation: generate small *error polynomials* s, e

$$t = a \cdot s + e$$

$$\text{pk} = (a, t), \ \text{sk} = s$$

Encryption: generate small *error polynomials* r, e_1, e_2

$$c_1 = a \cdot r + e_1$$

$$c_2 = t \cdot r + e_2 + m$$

Decryption: $m \approx c_2 - s \cdot c_1$

Lattice-based Encryption (LPR, NewHope, Kyber, . . .)

"Noisy ElGamal" with polynomials in $\mathbb{Z}_q[x]/\langle x^n + 1\rangle$

Key Generation: generate small *error polynomials* s, e

$t = a \cdot s + e$

$\text{pk} = (a, t), \ \text{sk} = s$

Encryption: generate small *error polynomials* r, e_1, e_2

$c_1 = a \cdot r + e_1$

$c_2 = t \cdot r + e_2 + m$

Decryption: $m \approx c_2 - s \cdot c_1$

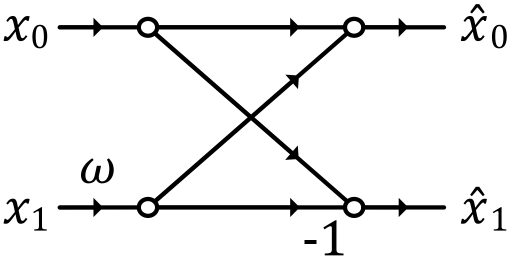
Number Theoretic Transform

- Naive polynomial multiplication: $\mathcal{O}(n^2)$
- Better: **N**umber **T**heoretic **T**ransform (NTT)
  - $\approx$ FFT in $\mathbb{Z}_q[x]$, runtime $\mathcal{O}(n \log n)$
  - pointwise mult. of NTT-transformed: $a \cdot b = \mathrm{INTT}(\mathrm{NTT}(a) \circ \mathrm{NTT}(b))$

Number Theoretic Transform

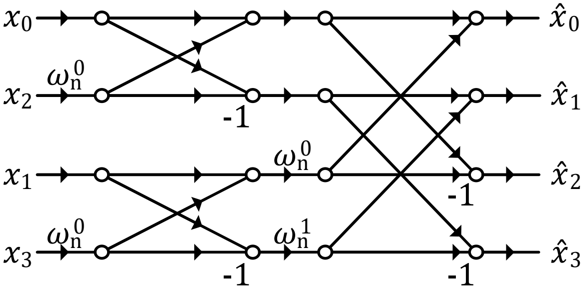
- Naive polynomial multiplication: $\mathcal{O}(n^2)$
- Better: **N**umber **T**heoretic **T**ransform (NTT)
  - $\approx$ FFT in $\mathbb{Z}_q[x]$, runtime $\mathcal{O}(n \log n)$
  - pointwise mult. of NTT-transformed: $a \cdot b = \mathrm{INTT}(\mathrm{NTT}(a) \circ \mathrm{NTT}(b))$

Butterfly



Butterfly = 2-coefficient NTT

Butterfly Network

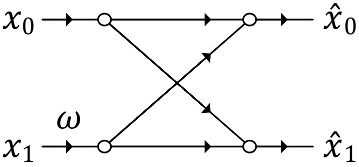


4-coefficient NTT

Previous Single-Trace Attack on the NTT

Recover secret NTT input with:

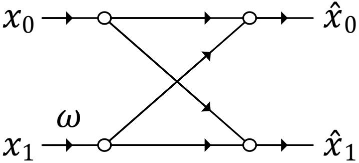
1. Template matching
   - Profile power consumption of mult.
   - Match profiles (templates) for probability distribution



Previous Single-Trace Attack on the NTT

Recover secret NTT input with:

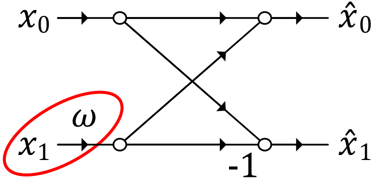
1. Template matching
   - Profile power consumption of mult.
   - Match profiles (templates) for probability distribution



Previous Single-Trace Attack on the NTT

Recover secret NTT input with:

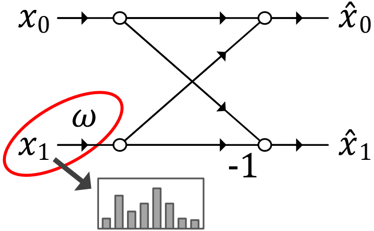
1. Template matching
   - Profile power consumption of mult.
   - Match profiles (templates) for probability distribution



Previous Single-Trace Attack on the NTT

Recover secret NTT input with:

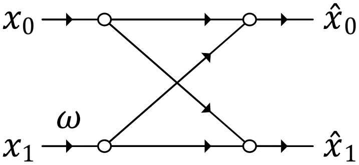
1. Template matching
   - Profile power consumption of mult.
   - Match profiles (templates) for probability distribution



Previous Single-Trace Attack on the NTT

Recover secret NTT input with:

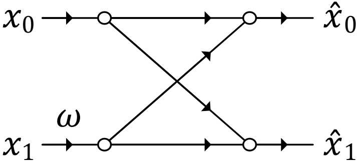
2. Belief propagation
   - Represent NTT with a graphical model
   - Pass beliefs along edges and update
   - Repeat until convergence reached



Previous Single-Trace Attack on the NTT

Recover secret NTT input with:

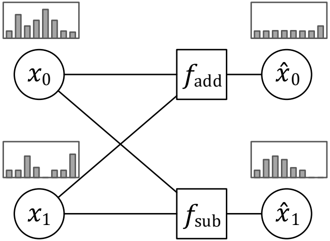
2. Belief propagation
   - Represent NTT with a graphical model
   - Pass beliefs along edges and update
   - Repeat until convergence reached



Previous Single-Trace Attack on the NTT

Recover secret NTT input with:

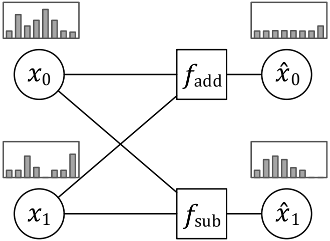
2. Belief propagation
   - Represent NTT with a graphical model
   - Pass beliefs along edges and update
   - Repeat until convergence reached



Previous Single-Trace Attack on the NTT

Recover secret NTT input with:

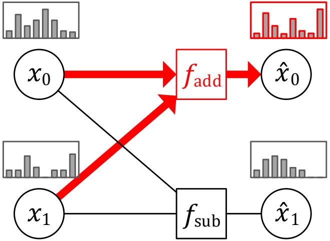
2. Belief propagation
   - Represent NTT with a graphical model
   - Pass beliefs along edges and update
   - Repeat until convergence reached



Previous Single-Trace Attack on the NTT

Recover secret NTT input with:

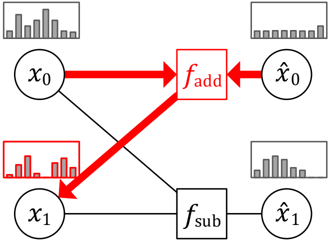
2. Belief propagation
   - Represent NTT with a graphical model
   - Pass beliefs along edges and update
   - Repeat until convergence reached



Previous Single-Trace Attack on the NTT

Recover secret NTT input with:

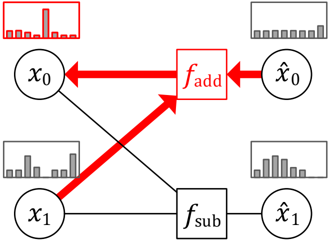
2. Belief propagation
   - Represent NTT with a graphical model
   - Pass beliefs along edges and update
   - Repeat until convergence reached



Previous Single-Trace Attack on the NTT

Recover secret NTT input with:

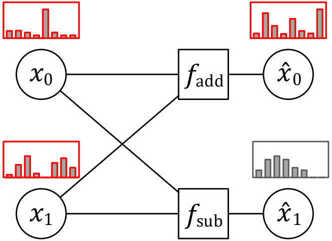
2. Belief propagation
    - Represent NTT with a graphical model
    - Pass beliefs along edges and update
    - Repeat until convergence reached



Previous Single-Trace Attack on the NTT

Recover secret NTT input with:

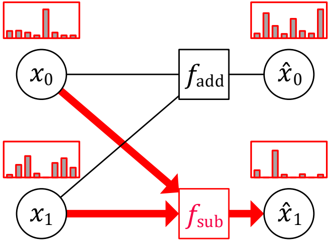
2. Belief propagation
   - Represent NTT with a graphical model
   - Pass beliefs along edges and update
   - Repeat until convergence reached



Previous Single-Trace Attack on the NTT

Recover secret NTT input with:

2. Belief propagation
   - Represent NTT with a graphical model
   - Pass beliefs along edges and update
   - Repeat until convergence reached



Practicality?

- Evaluation on non-constant-time implementation
  - timing information not needed per se
  - . . . but still aids attacks
- Requires powerful attacker
  - $\approx$ 1 million input combinations for modular multiplication
  - each one requires multivariate template
  - . . . very high templating effort

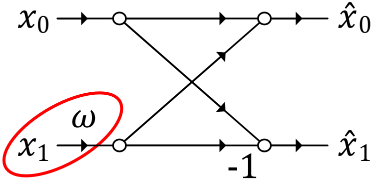
Practicality?

- Evaluation on non-constant-time implementation
  - timing information not needed per se
  - . . . but still aids attacks
- Requires powerful attacker
  - $\approx$ 1 million input combinations for modular multiplication
  - each one requires multivariate template
  - . . . very high templating effort

Decreased Templating Effort

Decreased Templating Effort

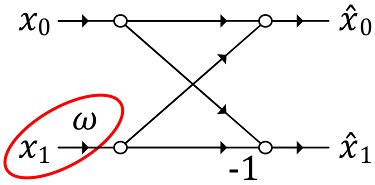
Previously



Target multiplication
1 million multivariate templates

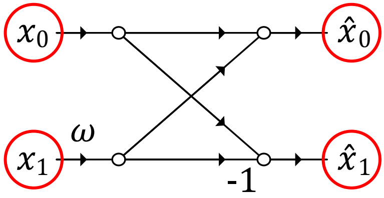
Decreased Templating Effort

Previously



Target multiplication
1 million multivariate templates

Now



Target memory loads and stores
14 univariate Hamming-weight templates

Are we done?

Are we done?

no timing information + simpler templates

↓

attack fails!

Changing Targets

Decryption

$m \approx c_2 - \mathrm{INTT}(\mathrm{NTT}(s) \circ \mathrm{NTT}(c_1))$

Recover INTT input, compute s

INTT input: $[0, q-1]^n$

Encryption

$c_1 = \mathrm{INTT}(\mathrm{NTT}(a) \circ \mathrm{NTT}(r)) + e_1$

Recover r, compute $m \approx c_2 - t \cdot r$

r is *small*: e.g., $[-2, 2]^n$

Changing Targets

Decryption

$m \approx c_2 - \mathrm{INTT}(\mathrm{NTT}(s) \circ \mathrm{NTT}(c_1))$

Recover INTT input, compute s

INTT input: $[0, q-1]^n$

Encryption

$c_1 = \mathrm{INTT}(\mathrm{NTT}(a) \circ \mathrm{NTT}(r)) + e_1$

Recover r, compute $m \approx c_2 - t \cdot r$

r is *small*: e.g., $[-2, 2]^n$

Changing Targets

Decryption

$m \approx c_2 - \text{INTT}(\text{NTT}(s) \circ \text{NTT}(c_1))$

Recover INTT input, compute s

INTT input: $[0, q-1]^n$

Encryption

$c_1 = \text{INTT}(\text{NTT}(a) \circ \text{NTT}(r)) + e_1$

Recover r, compute $m \approx c_2 - t \cdot r$

r is *small*: e.g., $[-2, 2]^n$

Changing Targets

Decryption

$m \approx c_2 - \mathrm{INTT}(\mathrm{NTT}(s) \circ \mathrm{NTT}(c_1))$

Recover INTT input, compute s

INTT input: $[0, q-1]^n$

Encryption

$c_1 = \mathrm{INTT}(\mathrm{NTT}(a) \circ \mathrm{NTT}(r)) + e_1$

Recover r, compute $m \approx c_2 - t \cdot r$

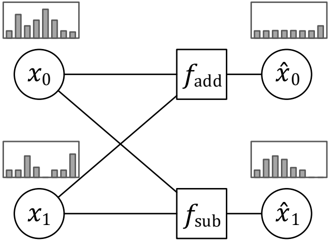
r is *small*: e.g., $[-2, 2]^n$

Attack simulations already work, but we can do better...

Belief Propagation and Loops

Information flow:

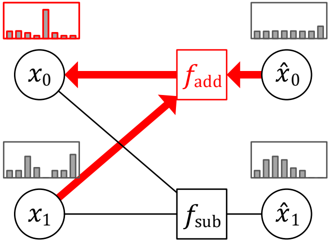
- $x_1 \longrightarrow x_0$
- $x_0 \longrightarrow x_1$
- Positive feedback loop
  - overconfidence, non-covergence
  - short loop, deterministic operations



Belief Propagation and Loops

Information flow:

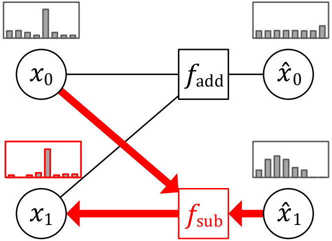
- $x_1 \rightarrow x_0$
- $x_0 \rightarrow x_1$
- Positive feedback loop
  - overconfidence, non-covergence
  - short loop, deterministic operations



Belief Propagation and Loops

Information flow:

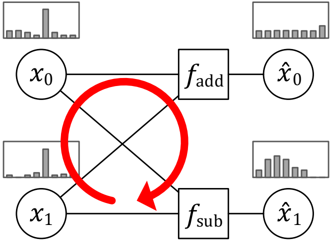
- $x_1 \rightarrow x_0$
- $x_0 \rightarrow x_1$
- Positive feedback loop
  - overconfidence, non-covergence
  - short loop, deterministic operations



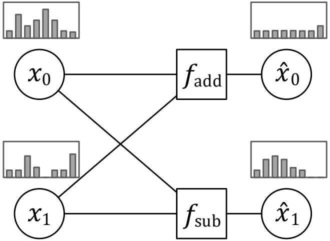
Belief Propagation and Loops

Information flow:

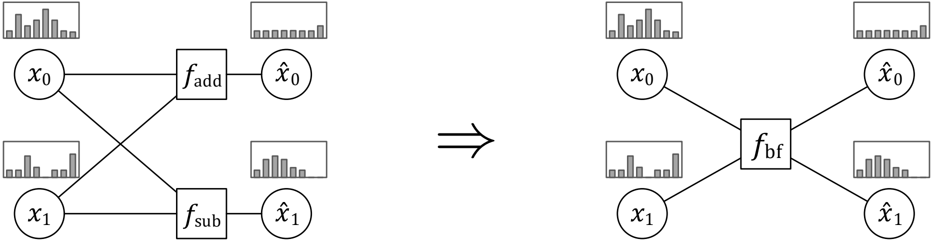
- $x_1 \rightarrow x_0$
- $x_0 \rightarrow x_1$
- Positive feedback loop
  - overconfidence, non-covergence
  - short loop, deterministic operations



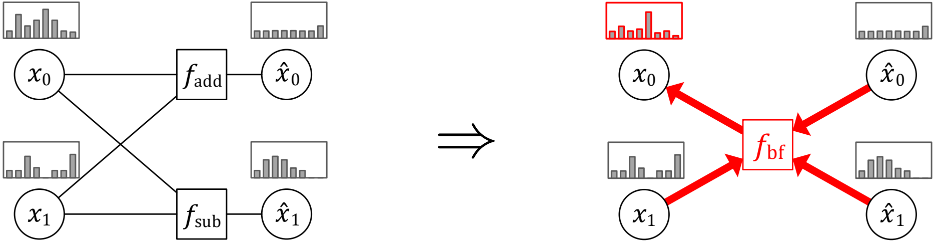
Butterfly Factors



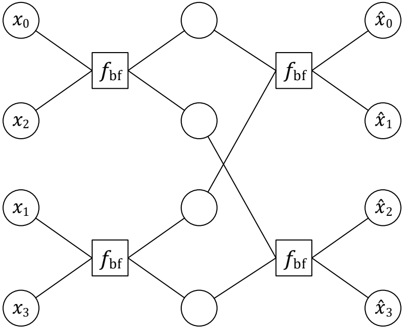
Butterfly Factors



Butterfly Factors

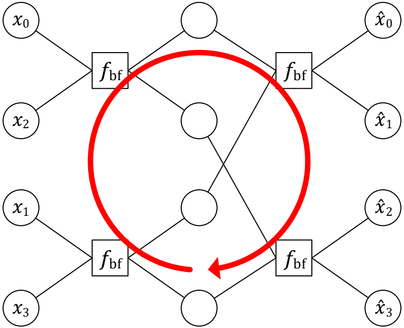


Still. . .



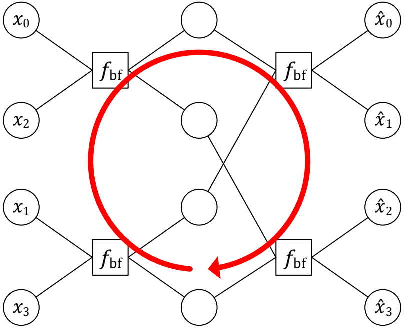
NTT with 4 coefficients

Still. . .



NTT with 4 coefficients

Still. . .

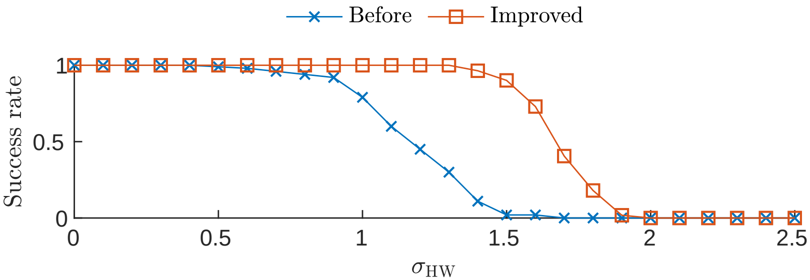


NTT with 4 coefficients

Still, shortest loops eliminated

Attack Simulations

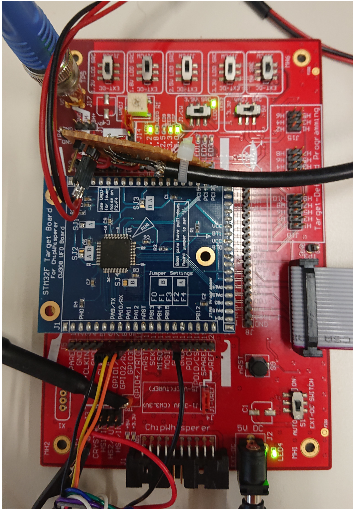
- Leakage simulations
  - Hamming-weight with Gaussian noise
- Tripling of σ^2 (SNR)



Attacking a Real Device

Power Analysis of an ARM Cortex M4

- ASM-optimized constant-time Kyber
- Profiling: 213 univariate HW templates
- Attack: matching and run BP
- Lattice reduction for error correction
- Overall success rate: 95%



More Results

- Analyzed masking countermeasure
  - adaptation required
  - attacks still possible, but at much lower noise
- Analysis of implementation techniques
  - lazy reductions, larger input ranges
  - reflect implementation techniques in graph

More Results

- Analyzed masking countermeasure
  - adaptation required
  - attacks still possible, but at much lower noise
- Analysis of implementation techniques
  - lazy reductions, larger input ranges
  - reflect implementation techniques in graph

More Results

- Analyzed masking countermeasure
  - adaptation required
  - attacks still possible, but at much lower noise
- Analysis of implementation techniques
  - lazy reductions, larger input ranges
  - reflect implementation techniques in graph

SCIENCE
PASSION
TECHNOLOGY

More Practical Single-Trace Attacks on the Number Theoretic Transform

Peter Pessl, Robert Primas
Graz University of Technology

LATINCRYPT 2019, October 02