Masking the GLP Lattice-Based Signature Scheme at Any Order

Sonia Belaïd

Joint Work with Gilles Barthe, Thomas Espitau, Pierre-Alain Fouque, Benjamin Grégoire, Mélissa Rossi, and Mehdi Tibouchi

CryptoExperts

Post-Quantum Schemes

Context

- NIST postquantum competition
- demands for practical implementation

Lattice-Based Signatures

So far

- Numerous physical attacks against lattice-based schemes (Gaussian distributions, rejection sampling)
- Few countermeasures exist

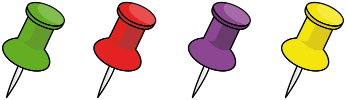
Power Analysis Attacks

Masking

- sound countermeasure which splits every sensitive variable x into $d+1$ shares $(x_i)_{0\leq i\leq d}$ such that
  - for every $1 \leq i \leq d$, x_i is picking uniformly at random
  - $x_0 \leftarrow x \oplus x_1 \oplus \cdots \oplus x_d$
- any strict subvector of at most d shares is independent from x
- d is called *masking order* or *security order*

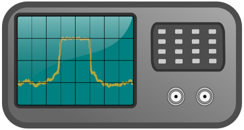
Leakage Models

- Probing model by Ishai, Sahai, and Wagner (Crypto 2003)
  - a circuit is d-probing secure iff any set composed of the exact values of at most d intermediate variables is independent from the secret



Leakage Models

- Probing model by Ishai, Sahai, and Wagner (Crypto 2003)
  - a circuit is d-probing secure iff any set composed of the exact values of at most d intermediate variables is independent from the secret
- Noisy leakage model by Chari, Jutla, Rao, and Rohatgi (Crypto 1999) then Rivain and Prouff (EC 2013)
  - a circuit is secure in the noisy leakage model iff the adversary cannot recover information on the secret from the noisy values of all the intermediate variables



Leakage Models

- Probing model by Ishai, Sahai, and Wagner (Crypto 2003)
  - a circuit is d-probing secure iff any set composed of the exact values of at most d intermediate variables is independent from the secret
- Noisy leakage model by Chari, Jutla, Rao, and Rohatgi (Crypto 1999) then Rivain and Prouff (EC 2013)
  - a circuit is secure in the noisy leakage model iff the adversary cannot recover information on the secret from the noisy values of all the intermediate variables
- Reduction by Duc, Dziembowski, and Faust (EC 2014)
  - d-probing security $\Rightarrow$ security in the noisy leakage model for some level of noise

Probing Model

- variables: secret, shares, constant
- masking order $d = 3$

function Ex-t3(x_0, x_1, x_2, x_3, c):

(* $x_0, x_1, x_2 = \$$ *)
(* $x_3 = x + x_0 + x_1 + x_2$ *)

$$r_0 \leftarrow \$$$
$$r_1 \leftarrow \$$$
$$y_0 \leftarrow x_0 + r_0$$
$$y_1 \leftarrow x_3 + r_1$$
$$t_1 \leftarrow x_1 + r_0$$
$$t_2 \leftarrow (x_1 + r_0) + x_2$$
$$y_2 \leftarrow (x_1 + r_0 + x_2) + r_1$$
$$y_3 \leftarrow c + r_1$$

return(y_0, y_1, y_2, y_3)

Probing Model

- variables: secret, shares, constant
- masking order $d = 3$

---

function Ex-t3(x_0, x_1, x_2, x_3, c):

(* $x_0, x_1, x_2 = \$$ *)
(* $x_3 = x + x_0 + x_1 + x_2$ *)

$r_0 \leftarrow \$$
$r_1 \leftarrow \$$
$y_0 \leftarrow x_0 + r_0$
$y_1 \leftarrow x_3 + r_1$
$t_1 \leftarrow x_1 + r_0$
$t_2 \leftarrow (x_1 + r_0) + x_2$
$y_2 \leftarrow (x_1 + r_0 + x_2) + r_1$
$y_3 \leftarrow c + r_1$
return(y_0, y_1, y_2, y_3)

---

independent from the secret? (r_1, t_2, y_3)

Probing Model

- variables: secret, shares, constant
- masking order $d = 3$

independent from the secret?

function Ex-t3(x_0, x_1, x_2, x_3, c):

(* $x_0, x_1, x_2 = \$$ *)
(* $x_3 = x + x_0 + x_1 + x_2$ *)

$r_0 \leftarrow \$$
$r_1 \leftarrow \$$
$y_0 \leftarrow x_0 + r_0$
$y_1 \leftarrow x_3 + r_1$
$t_1 \leftarrow x_1 + r_0$
$t_2 \leftarrow (x_1 + r_0) + x_2$
$y_2 \leftarrow (x_1 + r_0 + x_2) + r_1$
$y_3 \leftarrow c + r_1$

return(y_0, y_1, y_2, y_3)

Non-Interference (NI)

- d-NI $\Rightarrow$ d-probing secure
- a circuit is d-NI iff any set of d intermediate variables can be perfectly simulated with at most d shares of each input

can be simulated with x_0 and x_1

function Ex-t3(x_0, x_1, x_2, x_3, c):

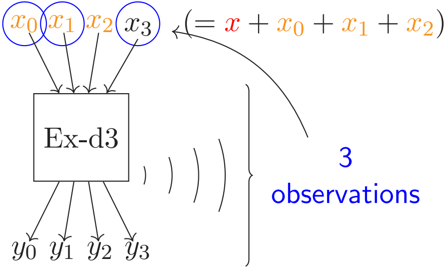
(* $x_0, x_1, x_2 = \$$ *)
(* $x_3 = x + x_0 + x_1 + x_2$ *)

$r_0 \leftarrow \$$
$r_1 \leftarrow \$$
$y_0 \leftarrow x_0 + r_0$
$y_1 \leftarrow x_3 + r_1$
$t_1 \leftarrow x_1 + r_0$
$t_2 \leftarrow (x_1 + r_0) + x_2$
$y_2 \leftarrow (x_1 + r_0 + x_2) + r_1$
$y_3 \leftarrow c + r_1$

return(y_0, y_1, y_2, y_3)

Non-Interference (NI)

- d-NI $\Rightarrow$ d-probing secure
- a circuit is d-NI iff any set of d intermediate variables can be perfectly simulated with at most d shares of each input



Strong Non-Interference (SNI)

- d-SNI $\Rightarrow$ d-NI $\Rightarrow$ d-probing secure
- a circuit is d-SNI iff any set of d intermediate variables, whose d_1 on the internal variables and d_2 and the outputs, can be perfectly simulated with at most d_1 shares of each input

function Ex-t3(x_0, x_1, x_2, x_3, c):

(* $x_0, x_1, x_2 = \$$ *)

(* $x_3 = x + x_0 + x_1 + x_2$ *)

$r_0 \leftarrow \$$

$r_1 \leftarrow \$$

$y_0 \leftarrow x_0 + r_0$

$y_1 \leftarrow x_3 + r_1$

$t_1 \leftarrow x_1 + r_0$

$t_2 \leftarrow (x_1 + r_0) + x_2$

$y_2 \leftarrow (x_1 + r_0 + x_2) + r_1$

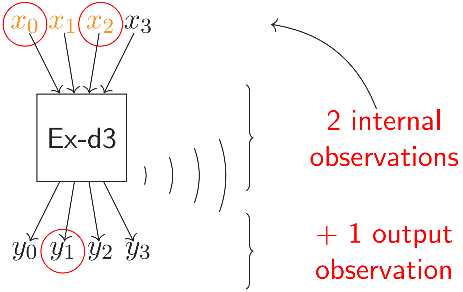
$y_3 \leftarrow c + r_1$

return(y_0, y_1, y_2, y_3)

require x_0 and x_1 to be perfectly simulated $\Rightarrow$ not 3-SNI since y_0 is an output variable

Strong Non-Interference (SNI)

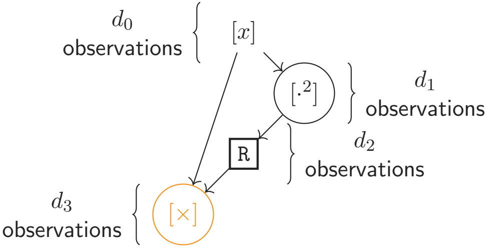
- d-SNI $\Rightarrow$ d-NI $\Rightarrow$ d-probing secure
- a circuit is d-SNI iff any set of d intermediate variables, whose d_1 on the internal variables and d_2 and the outputs, can be perfectly simulated with at most d_1 shares of each input



Composition

- From NI and SNI gadgets, we are able to build a NI circuit by adding SNI refresh gadgets when necessary
- Example: AES S-box on $\mathrm{GF}(2^8)$

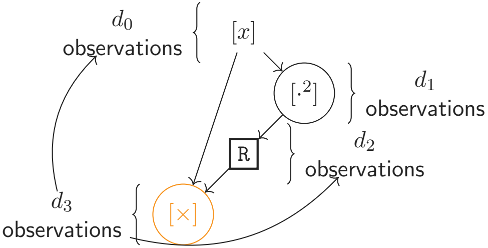
Constraint:
$d_0 + d_1 + d_2 + d_3 \leqslant d$



Composition

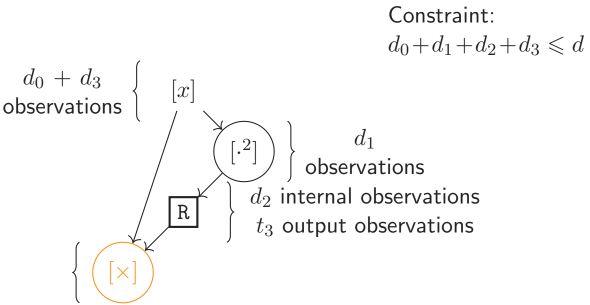
- From NI and SNI gadgets, we are able to build a NI circuit by adding SNI refresh gadgets when necessary
- Example: AES S-box on $\mathrm{GF}(2^8)$

Constraint:
$d_0 + d_1 + d_2 + d_3 \leqslant d$



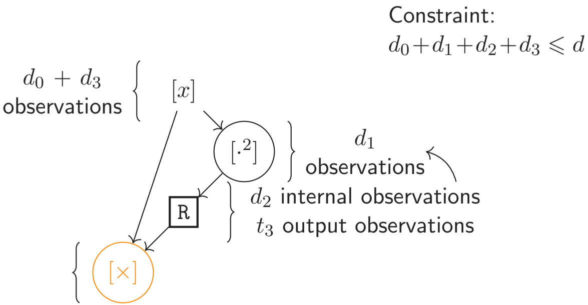
Composition

- From NI and SNI gadgets, we are able to build a NI circuit by adding SNI refresh gadgets when necessary
- Example: AES S-box on $\mathrm{GF}(2^8)$



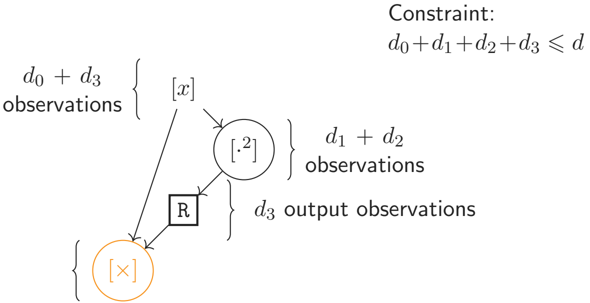
Composition

- From NI and SNI gadgets, we are able to build a NI circuit by adding SNI refresh gadgets when necessary
- Example: AES S-box on $\mathrm{GF}(2^8)$



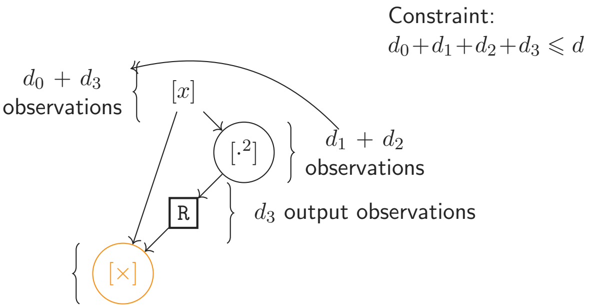
Composition

- From NI and SNI gadgets, we are able to build a NI circuit by adding SNI refresh gadgets when necessary
- Example: AES S-box on $\mathrm{GF}(2^8)$



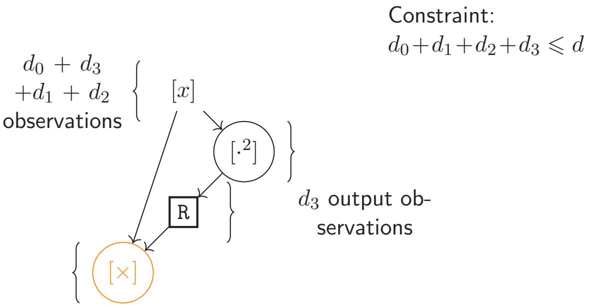
Composition

- From NI and SNI gadgets, we are able to build a NI circuit by adding SNI refresh gadgets when necessary
- Example: AES S-box on $\mathrm{GF}(2^8)$



Composition

- From NI and SNI gadgets, we are able to build a NI circuit by adding SNI refresh gadgets when necessary
- Example: AES S-box on $\mathrm{GF}(2^8)$



GLP Features

- Introduced at CHES 2012 by Tim Güneysu, Vadim Lyubashevsky, and Thomas Pöppelmann
- No Gaussians, only uniform distributions
- Probabilistic algorithm
- Rejection sampling

GLP Key Derivation

$$\mathcal{R} = \frac{\mathbb{Z}_p[x]}{x^n+1}$$

$\mathcal{R}_k$: coefficients in the range $[-k, k]$

Algorithm 1 GLP key derivation

Ensure: Signing key sk, verification key pk

1: $\mathbf{s_1}, \mathbf{s_2} \xleftarrow{\$} \mathcal{R}_1$ //$\mathbf{s_1}$ and $\mathbf{s_2}$ have coefficients in $\{-1, 0, 1\}$
2: $\mathbf{a} \xleftarrow{\$} \mathcal{R}$
3: $\mathbf{t} \leftarrow \mathbf{as_1} + \mathbf{s_2}$
4: $sk \leftarrow (\mathbf{s_1}, \mathbf{s_2})$
5: $pk \leftarrow (\mathbf{a}, \mathbf{t})$

- Based on the **Decisional Compact Knapsack** problem

GLP Signature

- Fiat–Shamir with abort signature

Algorithm 2 GLP signature

Require: $\mathbf{m}$, pk, sk
Ensure: Signature σ

1: $\mathbf{y}_1, \mathbf{y}_2 \xleftarrow{\$} \mathcal{R}_k$ — Random generation
2: $\mathbf{c} \leftarrow H(\mathbf{r} = \mathbf{a}\mathbf{y_1} + \mathbf{y_2}, \mathbf{m})$ — Commitment
3: $\mathbf{z}_1 \leftarrow \mathbf{s}_1\mathbf{c} + \mathbf{y}_1$
4: $\mathbf{z}_2 \leftarrow \mathbf{s}_2\mathbf{c} + \mathbf{y}_2$
5: **if** $\mathbf{z}_1$ or $\mathbf{z}_2 \notin \mathcal{R}_{k-\alpha}$ **then** restart — Rejection Sampling
return $\sigma = (\mathbf{z}_1, \mathbf{z}_2, \mathbf{c})$

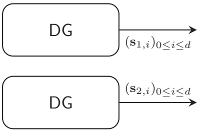
Verification $\mathbf{z}_1$, $\mathbf{z}_2 \in \mathcal{R}_{k-\alpha}$ and $\mathbf{c} = H(\mathbf{a}\mathbf{z}_1 + \mathbf{z}_2 - \mathbf{t}\mathbf{c}, \mathbf{m})$

Masking GLP Key Derivation

Algorithm 3 GLP key derivation

Ensure: Signing key sk, verification key pk

1: $\mathbf{s_1}, \mathbf{s_2} \xleftarrow{\$} \mathcal{R}_1$ //$\mathbf{s_1}$ and $\mathbf{s_2}$ have coefficients in $\{-1, 0, 1\}$
2: $\mathbf{a} \xleftarrow{\$} \mathcal{R}$
3: $\mathbf{t} \leftarrow \mathbf{a}\mathbf{s_1} + \mathbf{s_2}$
4: $sk \leftarrow (\mathbf{s_1}, \mathbf{s_2})$
5: $pk \leftarrow (\mathbf{a}, \mathbf{t})$

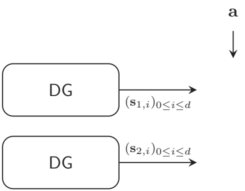


Masking GLP Key Derivation

Algorithm 4 GLP key derivation

Ensure: Signing key sk, verification key pk

1: $\mathbf{s_1}, \mathbf{s_2} \xleftarrow{\$} \mathcal{R}_1$ //$\mathbf{s_1}$ and $\mathbf{s_2}$ have coefficients in $\{-1, 0, 1\}$
2: $\mathbf{a} \xleftarrow{\$} \mathcal{R}$
3: $\mathbf{t} \leftarrow \mathbf{a}\mathbf{s_1} + \mathbf{s_2}$
4: $sk \leftarrow (\mathbf{s_1}, \mathbf{s_2})$
5: $pk \leftarrow (\mathbf{a}, \mathbf{t})$

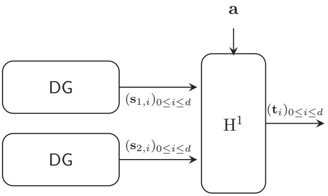


Masking GLP Key Derivation

Algorithm 5 GLP key derivation

Ensure: Signing key sk, verification key pk

1: $\mathbf{s_1}, \mathbf{s_2} \xleftarrow{\$} \mathcal{R}_1$ //$\mathbf{s_1}$ and $\mathbf{s_2}$ have coefficients in $\{-1, 0, 1\}$
2: $\mathbf{a} \xleftarrow{\$} \mathcal{R}$
3: $\mathbf{t} \leftarrow \mathbf{a}\mathbf{s_1} + \mathbf{s_2}$
4: $sk \leftarrow (\mathbf{s_1}, \mathbf{s_2})$
5: $pk \leftarrow (\mathbf{a}, \mathbf{t})$

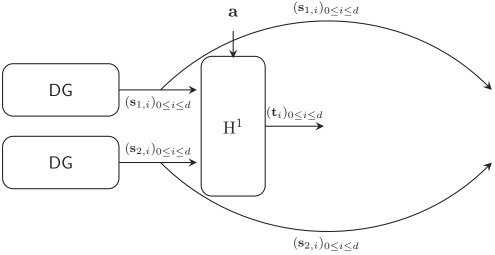


Masking GLP Key Derivation

Algorithm 6 GLP key derivation

Ensure: Signing key sk, verification key pk

1: $\mathbf{s_1}, \mathbf{s_2} \xleftarrow{\$} \mathcal{R}_1$ //$\mathbf{s_1}$ and $\mathbf{s_2}$ have coefficients in $\{-1, 0, 1\}$
2: $\mathbf{a} \xleftarrow{\$} \mathcal{R}$
3: $\mathbf{t} \leftarrow \mathbf{a}\mathbf{s_1} + \mathbf{s_2}$
4: $sk \leftarrow (\mathbf{s_1}, \mathbf{s_2})$
5: $pk \leftarrow (\mathbf{a}, \mathbf{t})$

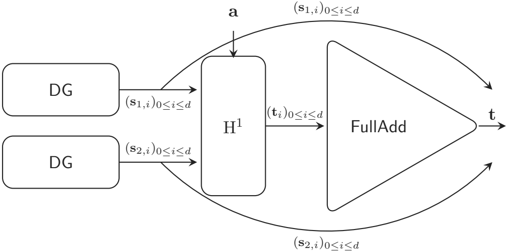


Masking GLP Key Derivation

Algorithm 7 GLP key derivation

Ensure: Signing key sk, verification key pk

1: $\mathbf{s_1}, \mathbf{s_2} \xleftarrow{\$} \mathcal{R}_1$ //$\mathbf{s_1}$ and $\mathbf{s_2}$ have coefficients in $\{-1, 0, 1\}$
2: $\mathbf{a} \xleftarrow{\$} \mathcal{R}$
3: $\mathbf{t} \leftarrow \mathbf{a}\mathbf{s_1} + \mathbf{s_2}$
4: $sk \leftarrow (\mathbf{s_1}, \mathbf{s_2})$
5: $pk \leftarrow (\mathbf{a}, \mathbf{t})$



Security of Individual Gadgets

- Each sensitive variable must be masked
  - mod-p arithmetic masking
  - w-bit Boolean masking

- Each block (or sub-block) which manipulates secret data is individually proven according to one of the three properties
  - Non-Interference (NI)
  - Strong Non-Interference (SNI)
  - Non-Interference with public Outputs (NIo)

Masking GLP Key Derivation

- DG: generation of sharings for coefficients $x \in [-k, k]$ $(k = 1)$

Masking GLP Key Derivation

- DG: generation of sharings for coefficients $x \in [-k, k]$ $(k = 1)$
  1. generate a Boolean sharing of x:

     $$\forall 0 \leq i \leq d, \quad x_i \leftarrow [0, 2^{w_0} - 1]$$

     where $2^{w_0} > 2k + 1 \geq 2^{w_0 - 1}$

Masking GLP Key Derivation

- DG: generation of sharings for coefficients $x \in [-k, k]$ $(k = 1)$
  1. generate a Boolean sharing of x:
     $$\forall 0 \leq i \leq d, \quad x_i \leftarrow [0, 2^{w_0} - 1]$$
     where $2^{w_0} > 2k + 1 \geq 2^{w_0 - 1}$
  2. $(\delta_i)_{0 \leq i \leq d} \leftarrow (\mathbf{x}_i)_{0 \leq i \leq d} - (\mathbf{k}_i)_{0 \leq i \leq d}$

Masking GLP Key Derivation

- DG: generation of sharings for coefficients $x \in [-k, k]$ $(k = 1)$
  1. generate a Boolean sharing of x:

     $$\forall 0 \leq i \leq d, \quad x_i \leftarrow [0, 2^{w_0} - 1]$$

     where $2^{w_0} > 2k + 1 \geq 2^{w_0 - 1}$
  2. $(\delta_i)_{0 \leq i \leq d} \leftarrow (\mathbf{x}_i)_{0 \leq i \leq d} - (\mathbf{k}_i)_{0 \leq i \leq d}$
  3. $b \leftarrow$ unmask δ's most significant bit
  4. b equals 0 iff $x \geq 2k + 1$

Masking GLP Key Derivation

- DG: generation of sharings for coefficients $x \in [-k, k]$ $(k = 1)$
  1. generate a Boolean sharing of x:
  $$\forall 0 \leq i \leq d, \quad x_i \leftarrow [0, 2^{w_0} - 1]$$
  where $2^{w_0} > 2k + 1 \geq 2^{w_0 - 1}$
  2. $(\delta_i)_{0 \leq i \leq d} \leftarrow (\mathbf{x}_i)_{0 \leq i \leq d} - (\mathbf{k}_i)_{0 \leq i \leq d}$
  3. $b \leftarrow$ unmask δ's most significant bit
  4. b equals 0 iff $x \geq 2k + 1$
  5. convert $(\mathbf{x}_i)_{0 \leq i \leq d}$ to an arithmetic masking

Masking GLP Key Derivation

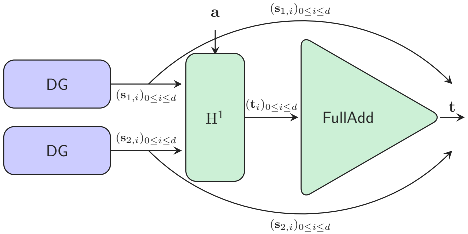
- DG: generation of sharings for coefficients $x \in [-k, k]$ $(k = 1)$
  1. generate a Boolean sharing of x:
  $$\forall 0 \leq i \leq d, \quad x_i \leftarrow [0, 2^{w_0} - 1]$$
  where $2^{w_0} > 2k + 1 \geq 2^{w_0 - 1}$
  2. $(\delta_i)_{0 \leq i \leq d} \leftarrow (\mathbf{x}_i)_{0 \leq i \leq d} - (\mathbf{k}_i)_{0 \leq i \leq d}$
  3. $b \leftarrow$ unmask δ's most significant bit
  4. b equals 0 iff $x \geq 2k + 1$
  5. convert $(\mathbf{x}_i)_{0 \leq i \leq d}$ to an arithmetic masking
- H^1: $\mathbf{t} \leftarrow \mathbf{a}\mathbf{s_1} + \mathbf{s_2}$
- FullAdd: refresh then add

Masking GLP Key Derivation

Algorithm 8 GLP key derivation

Ensure: Signing key sk, verification key pk

1: $\mathbf{s_1}, \mathbf{s_2} \xleftarrow{\$} \mathcal{R}_1$ //$\mathbf{s_1}$ and $\mathbf{s_2}$ have coefficients in $\{-1, 0, 1\}$
2: $\mathbf{a} \xleftarrow{\$} \mathcal{R}$
3: $\mathbf{t} \leftarrow \mathbf{a}\mathbf{s_1} + \mathbf{s_2}$
4: $sk \leftarrow (\mathbf{s_1}, \mathbf{s_2})$
5: $pk \leftarrow (\mathbf{a}, \mathbf{t})$



Not masked

Non interferent

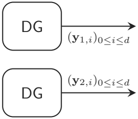
Non interferent with public outputs

Masking GLP Signature

Algorithm 9 GLP signature

Require: $\mathbf{m}$, pk, sk
Ensure: Signature σ

$\mathbf{y}_1, \mathbf{y}_2 \xleftarrow{\$} \mathcal{R}_k$
$\mathbf{c} \leftarrow H(\mathbf{r} = \mathbf{a}\mathbf{y_1} + \mathbf{y_2}, \mathbf{m})$
$\mathbf{z}_1 \leftarrow \mathbf{s}_1\mathbf{c} + \mathbf{y}_1$
$\mathbf{z}_2 \leftarrow \mathbf{s}_2\mathbf{c} + \mathbf{y}_2$
if $\mathbf{z}_1$ or $\mathbf{z}_2 \notin \mathcal{R}_{k-\alpha}$ **then** restart
return $\sigma = (\mathbf{z}_1, \mathbf{z}_2, \mathbf{c})$

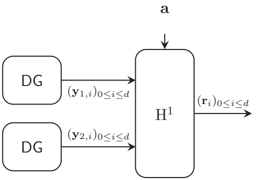


Masking GLP Signature

Algorithm 10 GLP signature

Require: $\mathbf{m}$, pk, sk
Ensure: Signature σ

1: $\mathbf{y}_1, \mathbf{y}_2 \xleftarrow{\$} \mathcal{R}_k$
2: $\mathbf{c} \leftarrow H(\mathbf{r} = \mathbf{a}\mathbf{y}_1 + \mathbf{y}_2, \mathbf{m})$
3: $\mathbf{z}_1 \leftarrow \mathbf{s}_1\mathbf{c} + \mathbf{y}_1$
4: $\mathbf{z}_2 \leftarrow \mathbf{s}_2\mathbf{c} + \mathbf{y}_2$
5: **if** $\mathbf{z}_1$ or $\mathbf{z}_2 \notin \mathcal{R}_{k-\alpha}$ **then** restart
return $\sigma = (\mathbf{z}_1, \mathbf{z}_2, \mathbf{c})$

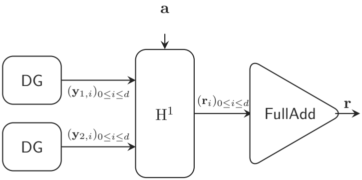


Masking GLP Signature

Algorithm 11 GLP signature

Require: $\mathbf{m}$, pk, sk
Ensure: Signature σ

1: $\mathbf{y}_1, \mathbf{y}_2 \xleftarrow{\$} \mathcal{R}_k$
2: $\mathbf{c} \leftarrow H(\mathbf{r} = \mathbf{a}\mathbf{y_1} + \mathbf{y_2}, \mathbf{m})$
3: $\mathbf{z}_1 \leftarrow \mathbf{s}_1\mathbf{c} + \mathbf{y}_1$
4: $\mathbf{z}_2 \leftarrow \mathbf{s}_2\mathbf{c} + \mathbf{y}_2$
5: **if** $\mathbf{z}_1$ or $\mathbf{z}_2 \notin \mathcal{R}_{k-\alpha}$ **then** restart
return $\sigma = (\mathbf{z}_1, \mathbf{z}_2, \mathbf{c})$

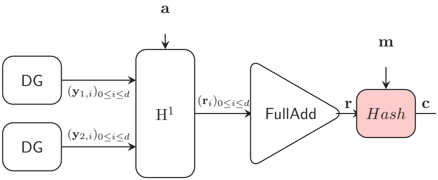


Masking GLP Signature

Algorithm 12 GLP signature

Require: $\mathbf{m}$, pk, sk
Ensure: Signature σ

1: $\mathbf{y}_1, \mathbf{y}_2 \xleftarrow{\$} \mathcal{R}_k$
2: $\mathbf{c} \leftarrow H(\mathbf{r} = \mathbf{a}\mathbf{y_1} + \mathbf{y_2}, \mathbf{m})$
3: $\mathbf{z}_1 \leftarrow \mathbf{s}_1\mathbf{c} + \mathbf{y}_1$
4: $\mathbf{z}_2 \leftarrow \mathbf{s}_2\mathbf{c} + \mathbf{y}_2$
5: **if** $\mathbf{z}_1$ or $\mathbf{z}_2 \notin \mathcal{R}_{k-\alpha}$ **then** restart
return $\sigma = (\mathbf{z}_1, \mathbf{z}_2, \mathbf{c})$

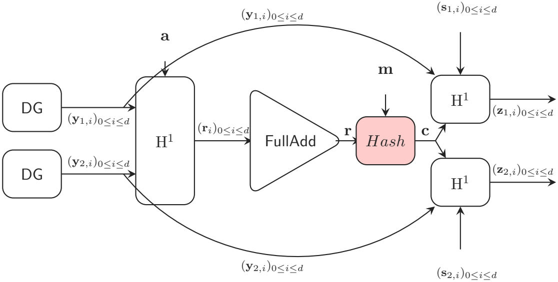


Masking GLP Signature

Algorithm 13 GLP signature

Require: $\mathbf{m}$, pk, sk
Ensure: Signature σ

1: $\mathbf{y}_1, \mathbf{y}_2 \xleftarrow{\$} \mathcal{R}_k$
2: $\mathbf{c} \leftarrow H(\mathbf{r} = \mathbf{a}\mathbf{y_1} + \mathbf{y_2}, \mathbf{m})$
3: $\mathbf{z}_1 \leftarrow \mathbf{s}_1\mathbf{c} + \mathbf{y}_1$
4: $\mathbf{z}_2 \leftarrow \mathbf{s}_2\mathbf{c} + \mathbf{y}_2$
5: **if** $\mathbf{z}_1$ or $\mathbf{z}_2 \notin \mathcal{R}_{k-\alpha}$ **then** restart
return $\sigma = (\mathbf{z}_1, \mathbf{z}_2, \mathbf{c})$

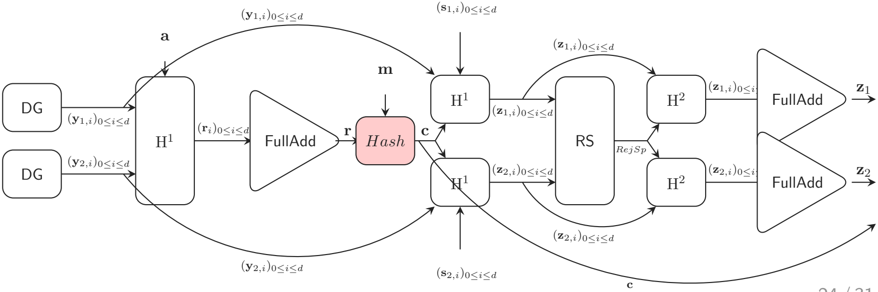


Masking GLP Signature

Algorithm 14 GLP signature

Require: $\mathbf{m}$, pk, sk
Ensure: Signature σ

1: $\mathbf{y}_1, \mathbf{y}_2 \xleftarrow{\$} \mathcal{R}_k$
2: $\mathbf{c} \leftarrow H(\mathbf{r} = \mathbf{a}\mathbf{y_1} + \mathbf{y_2}, \mathbf{m})$
3: $\mathbf{z}_1 \leftarrow \mathbf{s}_1\mathbf{c} + \mathbf{y}_1$
4: $\mathbf{z}_2 \leftarrow \mathbf{s}_2\mathbf{c} + \mathbf{y}_2$
5: **if** $\mathbf{z}_1$ or $\mathbf{z}_2 \notin \mathcal{R}_{k-\alpha}$ **then** restart
return $\sigma = (\mathbf{z}_1, \mathbf{z}_2, \mathbf{c})$



Masking GLP Signature

- DG: generation of sharings for coefficients $x \in [-k, k]$
- H^1: $\mathbf{a y_1 + y_2}$
- FullAdd: refresh then add
- $Hash$: unmasked
- RS: some details follow
- H^2: linear function
- FullAdd: refresh then add

Masking GLP Signature

- Rejection Sampling: are coefficients of $\mathbf{z_1}$ and $\mathbf{z_2}$ in $[-k+\alpha, k-\alpha]$?

Masking GLP Signature

- Rejection Sampling: are coefficients of $\mathbf{z_1}$ and $\mathbf{z_2}$ in $[-k+\alpha, k-\alpha]$?
  1. convert mod-p arithmetic sharing into Boolean masking

Masking GLP Signature

- Rejection Sampling: are coefficients of $\mathbf{z_1}$ and $\mathbf{z_2}$ in $[-k+\alpha, k-\alpha]$?
  1. convert mod-p arithmetic sharing into Boolean masking
  2. as in Data Generation, compute the masked difference with $k-\alpha$ difference

Masking GLP Signature

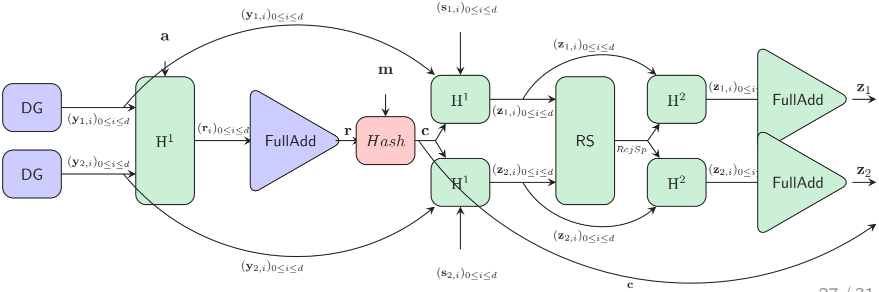
- Rejection Sampling: are coefficients of $\mathbf{z_1}$ and $\mathbf{z_2}$ in $[-k+\alpha, k-\alpha]$?
  1. convert mod-p arithmetic sharing into Boolean masking
  2. as in Data Generation, compute the masked difference with $k-\alpha$ difference
  3. securely check the most significant bit

Masking GLP Signature

Algorithm 15 GLP signature

Require: $\mathbf{m}$, pk, sk
Ensure: Signature σ

1: $\mathbf{y}_1, \mathbf{y}_2 \xleftarrow{\$} \mathcal{R}_k$
2: $\mathbf{c} \leftarrow H(\mathbf{r} = \mathbf{a}\mathbf{y_1} + \mathbf{y_2}, \mathbf{m})$
3: $\mathbf{z}_1 \leftarrow \mathbf{s}_1\mathbf{c} + \mathbf{y}_1$
4: $\mathbf{z}_2 \leftarrow \mathbf{s}_2\mathbf{c} + \mathbf{y}_2$
5: **if** $\mathbf{z}_1$ or $\mathbf{z}_2 \notin \mathcal{R}_{k-\alpha}$ **then** restart
return $\sigma = (\mathbf{z}_1, \mathbf{z}_2, \mathbf{c})$



Implementation

- unoptimized implementation
- based on a public domain implementation called GLYPH ($n = 1024$, $p = 59393$, $k = 16383$, and $\alpha = 16$)

Table: Implementation results. Timings are provided for 100 executions of the signing and verification algorithms, on one core of an Intel Core i7-3770 CPU-based desktop machine.

Number of shares $(d+1)$	Unprotected	2	3	4	5	6
Total CPU time (s)	0.540	8.15	16.4	39.5	62.1	111
Masking overhead	—	$\times 15$	$\times 30$	$\times 73$	$\times 115$	$\times 206$

Conclusion

In a nutshell...

- Higher-order masking of GLP with proof in the probing model
- New security notions to mask lattice-based signatures

To continue...

- Extend these results to other lattice-based signatures
- Extend these results to other post-quantum schemes