Trilithium: an efficient two-party ML-DSA protocol

Peeter Laud

with Antonín Dufka, Semjon Kravtšenko, Nikita Snetkov

SplitKey — server-supported RSA signatures

$$\mathrm{pk} = (n_1 n_2, e) \quad n_i = p_i q_i \quad d_i \cdot e \equiv 1 \pmod{\varphi(n_i)}$$
$$d'_1 + d''_1 \equiv d_1 \pmod{\varphi(n_1)} \quad u = \mathrm{Enc}(PIN, d'_1)$$

Client

u, n_1, n_2

Server

d''_1, d_2, n_1, n_2

SplitKey — server-supported RSA signatures

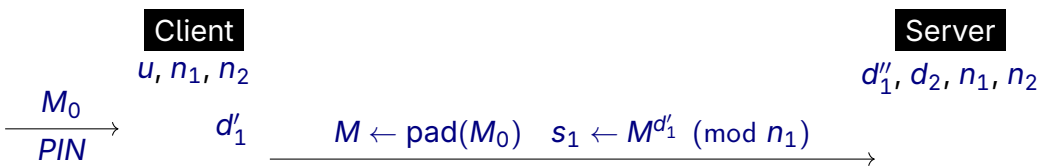
$$\mathrm{pk} = (n_1 n_2, e) \quad n_i = p_i q_i \quad d_i \cdot e \equiv 1 \pmod{\varphi(n_i)}$$
$$d'_1 + d''_1 \equiv d_1 \pmod{\varphi(n_1)} \quad u = \mathrm{Enc}(PIN, d'_1)$$

SplitKey — server-supported RSA signatures

$$\mathrm{pk} = (n_1 n_2, e) \quad n_i = p_i q_i \quad d_i \cdot e \equiv 1 \pmod{\varphi(n_i)}$$
$$d'_1 + d''_1 \equiv d_1 \pmod{\varphi(n_1)} \quad u = \mathrm{Enc}(PIN, d'_1)$$

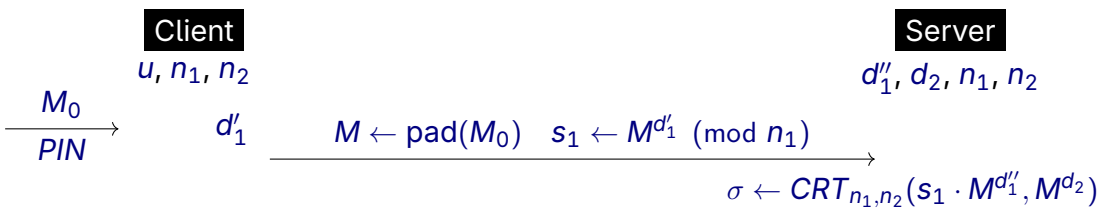
SplitKey — server-supported RSA signatures

$$\mathrm{pk} = (n_1 n_2, e) \quad n_i = p_i q_i \quad d_i \cdot e \equiv 1 \pmod{\varphi(n_i)}$$
$$d'_1 + d''_1 \equiv d_1 \pmod{\varphi(n_1)} \quad u = \mathrm{Enc}(PIN, d'_1)$$



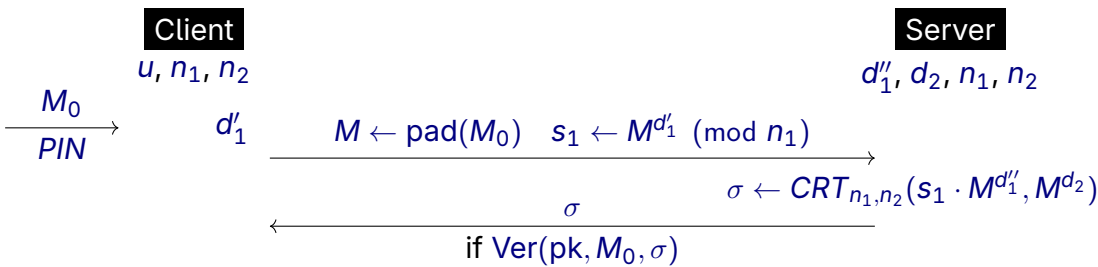
SplitKey — server-supported RSA signatures

$$\mathrm{pk} = (n_1 n_2, e) \quad n_i = p_i q_i \quad d_i \cdot e \equiv 1 \pmod{\varphi(n_i)}$$
$$d'_1 + d''_1 \equiv d_1 \pmod{\varphi(n_1)} \quad u = \mathrm{Enc}(PIN, d'_1)$$



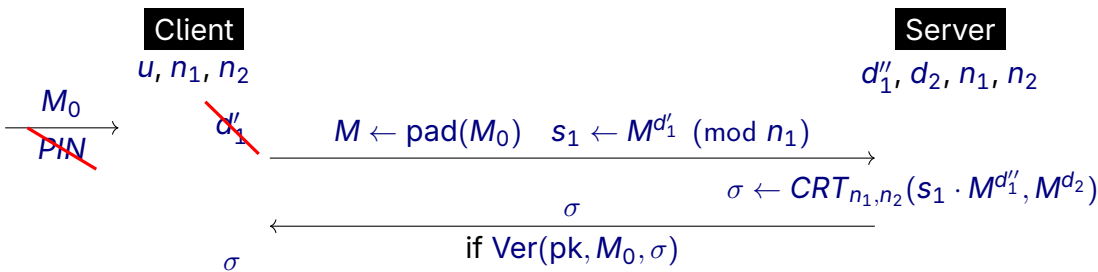
SplitKey — server-supported RSA signatures

$$\mathrm{pk} = (n_1 n_2, e) \quad n_i = p_i q_i \quad d_i \cdot e \equiv 1 \pmod{\varphi(n_i)}$$
$$d'_1 + d''_1 \equiv d_1 \pmod{\varphi(n_1)} \quad u = \mathrm{Enc}(PIN, d'_1)$$



SplitKey — server-supported RSA signatures

$$\mathsf{pk} = (n_1 n_2, e) \quad n_i = p_i q_i \quad d_i \cdot e \equiv 1 \pmod{\varphi(n_i)}$$
$$d'_1 + d''_1 \equiv d_1 \pmod{\varphi(n_1)} \quad u = \mathsf{Enc}(PIN, d'_1)$$



SplitKey deployment

- Client is implemented in a smartphone
  - If *u* leaks, then the attacker still cannot brute-force *PIN*
  - (a *clone detection* mechanism allows Server to reset the wrong PIN counter)
- Produces functionally interchangable RSA-PSS signatures
  - The length of keys and signatures is double the usual
  - Multi-primality of the modulus is not detected
- Smart-ID service based on SplitKey is running in Baltic states since 2016
  - 6.1M population. 3.7M users. www.smart-id.com
  - There are other, smaller deployments of SplitKey technology, too

SplitKey deployment

- Client is implemented in a smartphone
  - If u leaks, then the attacker still cannot brute-force *PIN*
  - (a *clone detection* mechanism allows Server to reset the wrong PIN counter)
- Produces functionally interchangable RSA-PSS signatures
  - The length of keys and signatures is double the usual
  - Multi-primality of the modulus is not detected
- Smart-ID service based on SplitKey is running in Baltic states since 2016
  - 6.1M population. 3.7M users. www.smart-id.com
  - There are other, smaller deployments of SplitKey technology, too

The days of Splitkey are numbered

Must replace RSA with some postquantum-secure signature scheme

Desiderata

- Two-party protocol
- “SplitKey-like properties”
  - Not in today’s talk. Should not be too difficult
- Standardized
  - At least *interchangable* with a standardized scheme
- Certifiable
  - Can be meaningfully made to pass the tests created for single-party key generation / signing

Dilithium / ML-DSA details

- $q \in \mathbb{P}$, $q = 2^{23} - 2^{13} + 1 = 2^{13} \cdot 3 \cdot 11 \cdot 31 + 1$. $R_q = \mathbb{Z}_q[X^N + 1]$, $N = 256$
- Think of $\mathbb{Z}_q$ as $\{-\lfloor \frac{q}{2} \rfloor, -\lfloor \frac{q}{2} \rfloor + 1, \ldots, -1, 0, 1, \ldots, \lfloor \frac{q}{2} \rfloor\}$
  - I.e. each element of $\mathbb{Z}_q$ has *size* between 0 and $\lfloor \frac{q}{2} \rfloor$
- $\alpha = (q-1)/44$ or $\alpha = (q-1)/16$
- "Size" generalizes coefficient- and component-wise to polynomials and tuples. Denote $\|p\|_\infty$, $\|\mathbf{p}\|_\infty$
- $S_\eta := \{p \in R_q \mid \|p\|_\infty \leq \eta\}$
- $B_\tau \subset S_1$; elements of B_τ have *exactly* τ non-zero coefficients

Let $x' = (x + \frac{\alpha}{2} - 1) \bmod q$
(i.e. $x' \in \{0, \ldots, q-1\}$)

$$x^H := \begin{cases} \lfloor x'/\alpha \rfloor, & x' \neq q-1 \\ 0, & x' = q-1 \end{cases}$$

$$x^L := x - \alpha \cdot x^H \pmod^{\pm} q$$

Simplified Dilithium / ML-DSA

Signing

- $\mathbf{y} \leftarrow\$ S^{\ell}_{\gamma_1 - 1}$
- $\mathbf{w} \leftarrow \mathbf{A} \cdot \mathbf{y}$
- $c \leftarrow H(M, \mathbf{w}^H) \in B_\tau$
- $\mathbf{z} \leftarrow \mathbf{y} + c \cdot \mathbf{s}_1$
- $\mathbf{x} \leftarrow \mathbf{w}^L - c \cdot \mathbf{s}_2$
- if $\|\mathbf{z}\|_\infty \geq \gamma_1 - \beta$, restart
- if $\|\mathbf{x}\|_\infty \geq \gamma_2 - \beta$, restart
- return $(\mathbf{z}, c)$

Key generation

- $\mathbf{A} \leftarrow\$ R_q^{k \times \ell}$ $\quad \mathbf{s}_1 \leftarrow\$ S_\eta^\ell$ $\quad \mathbf{s}_2 \leftarrow\$ S_\eta^k$
- $\mathbf{t} \leftarrow \mathbf{A} \cdot \mathbf{s}_1 + \mathbf{s}_2$
- $\mathsf{pk} = (\mathbf{A}, \mathbf{t})$ $\quad \mathsf{sk} = (\mathbf{s}_1, \mathbf{s}_2)$

Verification

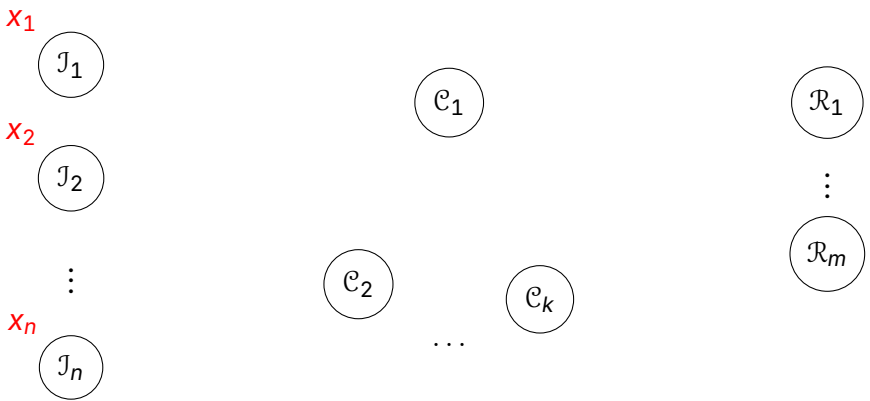
- $\mathbf{w}' \leftarrow \mathbf{A} \cdot \mathbf{z} - c \cdot \mathbf{t}$ Note that $\mathbf{w}' = \mathbf{x}$
- Check if $\|\mathbf{z}\|_\infty < \gamma_1 - \beta$ and $c = H(M, \mathbf{w}'^H)$

(The *decomposition parameter* α is $2\gamma_2$)

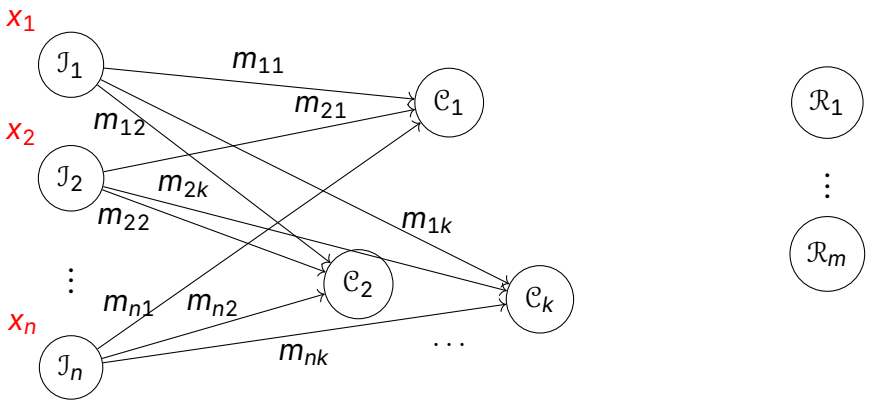
Parameter sizes

Parameters (see sections 5 and 6 of this document)	Values assigned by each parameter set ML-DSA-44	ML-DSA-65	ML-DSA-87
q - modulus [see §5]	8380417	8380417	8380417
d - # of dropped bits from $\mathbf{t}$ [see §5]	13	13	13
τ - # of ± 1's in polynomial c [see §6]	39	49	60
λ - collision strength of $\tilde{c}$ [see §6]	128	192	256
γ_1 - coefficient range of $\mathbf{y}$ [see §6]	2^{17}	2^{19}	2^{19}
γ_2 - low-order rounding range [see §6]	$(q-1)/88$	$(q-1)/32$	$(q-1)/32$
(k,ℓ) - dimensions of $\mathbf{A}$ [see §5]	(4,4)	(6,5)	(8,7)
η - private key range [see §5]	2	4	2
$\beta = \tau \cdot \eta$ [see §6]	78	196	120
ω - max # of 1's in the hint $\mathbf{h}$ [see §6]	80	55	75
Challenge entropy $\log \binom{256}{\tau} + \tau$ [see §6]	192	225	257
Repetitions (see explanation below)	4.25	5.1	3.85

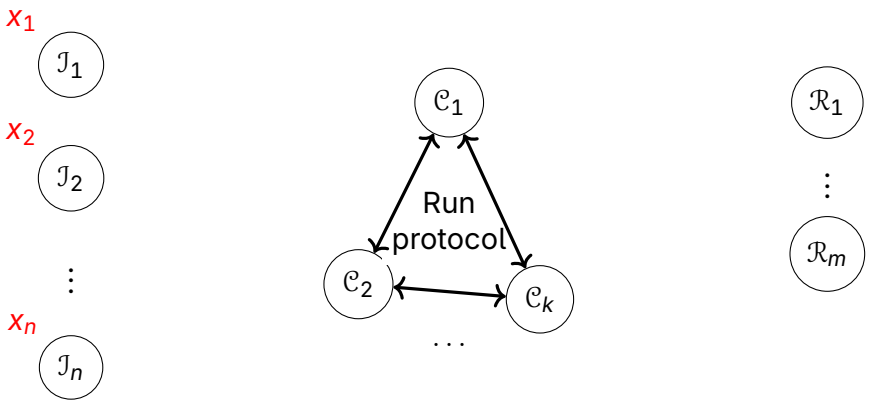
Secure multiparty computation



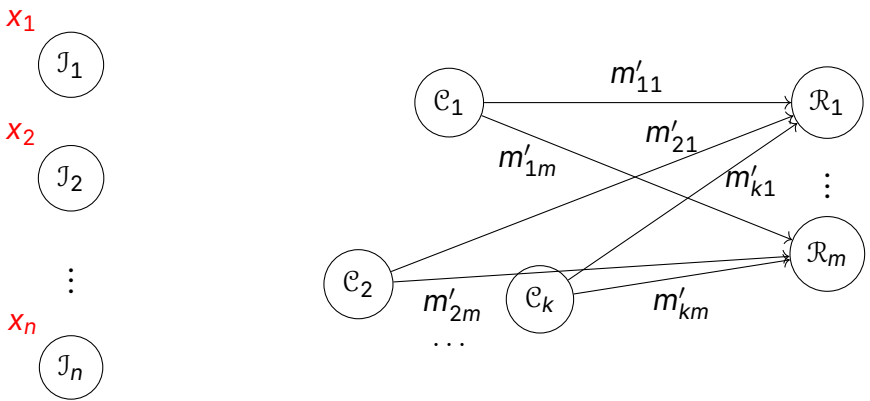
Secure multiparty computation



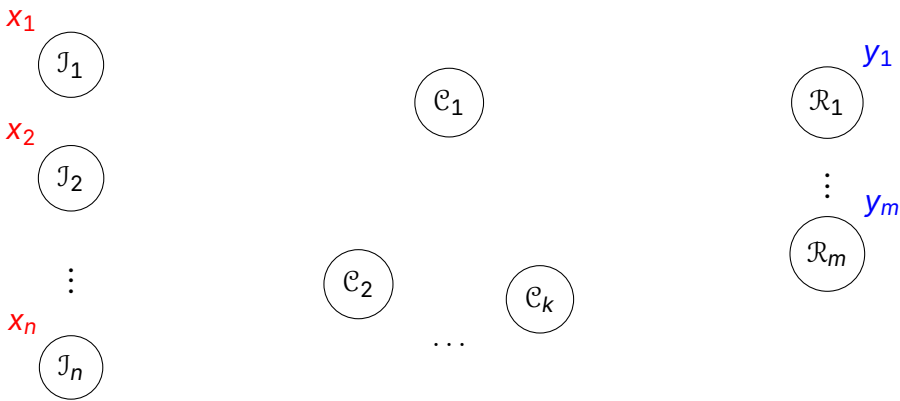
Secure multiparty computation



Secure multiparty computation



Secure multiparty computation



Public and private values

Inputs

- Some inputs from $\mathcal{I}_i$ may be public
  - It's OK if anyone learns them
  - Think each of $m_{i1}, \ldots, m_{ik}$ including such input
- Other inputs from $\mathcal{I}_i$ may be private
  - Think a part of $m_{i1}, \ldots, m_{ik}$ being a secret sharing of that input
  - $\mathcal{I}_i$ computes that sharing

Public and private values

Inputs

- Some inputs from $\mathcal{I}_i$ may be public
  - It's OK if anyone learns them
  - Think each of $m_{i1}, \ldots, m_{ik}$ including such input
- Other inputs from $\mathcal{I}_i$ may be private
  - Think a part of $m_{i1}, \ldots, m_{ik}$ being a secret sharing of that input
  - $\mathcal{I}_i$ computes that sharing

Outputs

- Also public or private
- $\mathcal{R}_j$ reconstructs its private output(s)

Public and private values

Inputs

- Some inputs from $\mathcal{I}_i$ may be public
  - It's OK if anyone learns them
  - Think each of $m_{i1}, \ldots, m_{ik}$ including such input
- Other inputs from $\mathcal{I}_i$ may be private
  - Think a part of $m_{i1}, \ldots, m_{ik}$ being a secret sharing of that input
  - $\mathcal{I}_i$ computes that sharing

Outputs

- Also public or private
- $\mathcal{R}_j$ reconstructs its private output(s)

Computation

- $\mathcal{C}_1, \ldots, \mathcal{C}_k$ run a protocol:
  - "shares of inputs" $\rightarrow$ "shares of outputs"
  - no leaks to a single $\mathcal{C}_i$
    - (or a small set of them)

$\mathcal{C}_1, \ldots, \mathcal{C}_k$ as a secure virtual machine

- Think that there's some program computing $y_1, \ldots, y_m$ from $x_1, \ldots, x_n$
  - A sequence of three-address operations…e.g. $a_1 \leftarrow \text{mul } a_2\ a_3$
    - …where all addresses are public
- For each possible operation, there is a protocol for $\mathcal{C}_1, \ldots, \mathcal{C}_k$:
  - "shares of inputs" $\longrightarrow$ "shares of the output"
- Predetermined addresses for inputs and outputs of the computation

$\mathcal{C}_1, \ldots, \mathcal{C}_k$ as a secure virtual machine

- Think that there's some program computing $y_1, \ldots, y_m$ from $x_1, \ldots, x_n$
  - A sequence of three-address operations…e.g. $a_1 \leftarrow \text{mul } a_2\ a_3$
    - …where all addresses are public
- For each possible operation, there is a protocol for $\mathcal{C}_1, \ldots, \mathcal{C}_k$:
  - "shares of inputs" $\longrightarrow$ "shares of the output"
- Predetermined addresses for inputs and outputs of the computation
- Program may branch on public values
  - $a_1 \leftarrow$ declassify a_2 is an available operation

Additive secret sharing

- A modulus $q \in \mathbb{Z}$ has been fixed
- The values are elements of $\mathbb{Z}_q$
  - $\{0, 1, \ldots, q-1\}$, with arithmetic modulo q
- Sharing $v \in \mathbb{Z}_q$ as **r**: pick random $r_1, \ldots, r_{k-1} \in \mathbb{Z}_q$, let $r_k = v - \sum_{i=1}^{k-1} r_i$
- Recovery of v from **r**: compute $v \leftarrow r_1 + \cdots + r_k$
  - …modulo q

Additive secret sharing

- A modulus $q \in \mathbb{Z}$ has been fixed
- The values are elements of $\mathbb{Z}_q$
  - $\{0, 1, \ldots, q-1\}$, with arithmetic modulo q
- Sharing $v \in \mathbb{Z}_q$ as **r**: pick random $r_1, \ldots, r_{k-1} \in \mathbb{Z}_q$, let $r_k = v - \sum_{i=1}^{k-1} r_i$
- Recovery of v from **r**: compute $v \leftarrow r_1 + \cdots + r_k$
  - …modulo q
- Let $[\![v]\!]$ denote "v that is held shared among $\mathcal{C}_1, \ldots, \mathcal{C}_k$"
  - Let $[\![v]\!]_i$ denote the share held by $\mathcal{C}_i$

Linear computations

- Given shared values $[\![x]\!], [\![y]\!]$, computing parties can compute:
  - $[\![x + y]\!]$, $[\![x - y]\!]$
  - $[\![c \cdot x]\!]$ for some constant $c \in \mathbb{Z}_q$
- These can be computed without the computing parties talking to each other
  - Computing party does the operations locally, on its share(s)
- Can also locally create $[\![c]\!]$ for a constant $c \in \mathbb{Z}_q$

Multiplication as a non-linear computation

- Have $[\![x]\!]$, $[\![y]\!]$. Want $[\![z]\!]$, where $z = xy \pmod q$
- Suppose computing parties have obtained $[\![a]\!], [\![b]\!], [\![c]\!]$, such that
  - $a, b \in \mathbb{Z}_q$ are generated uniformly randomly. $c = ab$
  - $[\![a]\!], [\![b]\!], [\![c]\!]$ have not been used anywhere in the computation

Protocol

- Compute $[\![\alpha]\!] = [\![x]\!] - [\![a]\!]$ and $[\![\beta]\!] = [\![y]\!] - [\![b]\!]$
- Declassify α and β. (This is privacy-preserving)
- Compute $[\![z]\!] \leftarrow \alpha \cdot [\![y]\!] + \beta \cdot [\![a]\!] + [\![c]\!]$

Creating $[\![a]\!], [\![b]\!], [\![c]\!]$

- An *offline* protocol, or
- a separate "Correlated Randomness Provider" (CRP) party

A typical pattern of MPC protocols

1. Compute some linear combinations of shared values
   - with constants that all computing parties know
2. Declassify some values
   - These become constants that all computing parties know
   - Make sure that declassification is privacy-preserving
     - E.g. they have been masked with randomness not used elsewhere
3. GOTO 1

Passive vs. active corruptions

- An (external) adversary may “corrupt” the parties
  - The protocol is secure as long as not too many parties are corrupted
- A passively corrupted party follows the protocol, but reports all it sees to the adversary
- An actively corrupted party also gets from the adversary the messages it sends to other parties
- There are surely contexts where *passively secure* MPC makes sense
  - Threshold computation of signatures is not one of these contexts
  - We’ll come back to it later

Easy and hard parts of signing under MPC

Signing

- $\mathbf{y} \leftarrow\$ S^{\ell}_{\gamma_1 - 1}$
- $\mathbf{w} \leftarrow \mathbf{A} \cdot \mathbf{y}$
- $c \leftarrow H(M, \mathbf{w}^H) \in B_\tau$
- $\mathbf{z} \leftarrow \mathbf{y} + c \cdot \mathbf{s}_1$
- $\mathbf{x}' \leftarrow \mathbf{w}^L - c \cdot \mathbf{s}_2$
- if $\|\mathbf{z}\|_\infty \geq \gamma_1 - \beta$, restart
- if $\|\mathbf{x}'\|_\infty \geq \gamma_2 - \beta$, restart
- return $(\mathbf{z}, c)$

Set-up

- Additive secret sharing modulo q
- $\mathbf{w}^H$ is declassified. c computed in clear

How?

- Green — linear computations
- Blue — we know how to do
  - Elements of $\mathbf{y}$ are between $-2^{\gamma_1 - 1}$ and $2^{\gamma_1 - 1} - 1$
    - Can generate random shared bits modulo q
- Red — complicated

Easy and hard parts of signing under MPC

Signing

- $\mathbf{y} \leftarrow\$ S^{\ell}_{\gamma_1 - 1}$
- $\mathbf{w} \leftarrow \mathbf{A} \cdot \mathbf{y}$
- $c \leftarrow H(M, \mathbf{w}^H) \in B_\tau$
- $\mathbf{z} \leftarrow \mathbf{y} + c \cdot \mathbf{s}_1$
- $\mathbf{x}' \leftarrow \mathbf{w}^L - c \cdot \mathbf{s}_2$
- if $\|\mathbf{z}\|_\infty \geq \gamma_1 - \beta$, restart
- if $\|\mathbf{x}'\|_\infty \geq \gamma_2 - \beta$, restart
- return $(\mathbf{z}, c)$

Set-up

- Additive secret sharing modulo q
- $\mathbf{w}^H$ is declassified. c computed in clear

How?

- Green — linear computations
- Blue — we know how to do
  - Elements of $\mathbf{y}$ are between $-2^{\gamma_1 - 1}$ and $2^{\gamma_1 - 1} - 1$
    - Can generate random shared bits modulo q
- Red — complicated

Intro to high bits: bit decomposition

Task

Given $[\![v]\!]$ (modulo q), find $[\![v_0]\!], \ldots, [\![v_{n-1}]\!]$ ($2^n > q$), such that:

- $v_0, \ldots, v_{n-1} \in \{0, 1\}$
- $\sum_{i=0}^{n-1} 2^i v_i = v$

Intro to high bits: bit decomposition

Task

Given $[\![v]\!]$ (modulo q), find $[\![v_0]\!], \ldots, [\![v_{n-1}]\!]$ ($2^n > q$), such that:

- $v_0, \ldots, v_{n-1} \in \{0, 1\}$
- $\sum_{i=0}^{n-1} 2^i v_i = v$

Correlated randomness

$[\![r]\!], [\![r_0]\!], \ldots, [\![r_{n-1}]\!]$

- $r \in \mathbb{Z}_q$ is random; rest is like v-s

Intro to high bits: bit decomposition

Task

Given $[\![v]\!]$ (modulo q), find $[\![v_0]\!], \ldots, [\![v_{n-1}]\!]$ ($2^n > q$), such that:

- $v_0, \ldots, v_{n-1} \in \{0, 1\}$
- $\sum_{i=0}^{n-1} 2^i v_i = v$

Correlated randomness

$[\![r]\!], [\![r_0]\!], \ldots, [\![r_{n-1}]\!]$

- $r \in \mathbb{Z}_q$ is random; rest is like v-s

Protocol

- Compute $y \leftarrow \text{declassify}([\![v]\!] - [\![r]\!])$
- Execute bitwise addition circuit:

$$\begin{array}{cccc} [\![r_{n-1}]\!] & \cdots & [\![r_1]\!] & [\![r_0]\!] \\ y_{n-1} & \cdots & y_1 & y_0 \\ \hline [\![v_{n-1}]\!] & \cdots & [\![v_1]\!] & [\![v_0]\!] \end{array}$$

Intro to high bits: bit decomposition

Task

Given $[\![v]\!]$ (modulo q), find $[\![v_0]\!], \ldots, [\![v_{n-1}]\!]$ ($2^n > q$), such that:

- $v_0, \ldots, v_{n-1} \in \{0, 1\}$
- $\sum_{i=0}^{n-1} 2^i v_i = v$

Correlated randomness

$[\![r]\!], [\![r_0]\!], \ldots, [\![r_{n-1}]\!]$

- $r \in \mathbb{Z}_q$ is random; rest is like v-s

Protocol

- Compute $y \leftarrow \text{declassify}([\![v]\!] - [\![r]\!])$
- Execute bitwise addition circuit:

$$\begin{array}{cccc} [\![r_{n-1}]\!] & \cdots & [\![r_1]\!] & [\![r_0]\!] \\ y_{n-1} & \cdots & y_1 & y_0 \\ \hline [\![v_{n-1}]\!] & \cdots & [\![v_1]\!] & [\![v_0]\!] \end{array}$$

Result

$\sum_{i=0}^{n-1} 2^i v_i \in \{v, v + q\}$

More general intro: characteristic vectors

- The characteristic vector of $v \in \{0, 1, \ldots, U-1\}$ is $(\underbrace{0, \ldots, 0}_{v \text{ times}}, 1, \underbrace{0, \ldots, 0}_{(U-v-1) \text{ times}})$
- Sometimes we represent a private v as $[\![v_0]\!], \ldots, [\![v_{U-1}]\!]$
  - Only if U is not too large
- This linearizes *all* functions $f : \{0, 1, \ldots, U-1\} \to Y$:

$$[\![f(v)]\!] = \sum_{i=0}^{U-1} f(i) \cdot [\![v_i]\!]$$

Computing the characteristic vector

Task

Given $[\![v]\!]$ (modulo q), find $[\![v_0]\!], \ldots, [\![v_{q-1}]\!]$, such that:

- $v_0, \ldots, v_{q-1}$ is the C.V. of v

(Note that length of vector equals sharing modulus)

Computing the characteristic vector

Task

Given $[\![v]\!]$ (modulo q), find $[\![v_0]\!], \ldots, [\![v_{q-1}]\!]$, such that:

- $v_0, \ldots, v_{q-1}$ is the C.V. of v

(Note that length of vector equals sharing modulus)

Correlated randomness

$[\![r]\!], [\![r_0]\!], \ldots, [\![r_{q-1}]\!]$

- $r \in \mathbb{Z}_q$ is random; rest is the C.V. of r

Computing the characteristic vector

Task

Given $[\![v]\!]$ (modulo q), find $[\![v_0]\!], \ldots, [\![v_{q-1}]\!]$, such that:

- $v_0, \ldots, v_{q-1}$ is the C.V. of v

(Note that length of vector equals sharing modulus)

Correlated randomness

$[\![r]\!], [\![r_0]\!], \ldots, [\![r_{q-1}]\!]$

- $r \in \mathbb{Z}_q$ is random; rest is the C.V. of r

Protocol

- Compute $y \leftarrow \text{declassify}([\![v]\!] - [\![r]\!])$
- Rotate **r**, so it becomes **v**:

$$[\![v_i]\!] \leftarrow [\![r_{(i-y) \bmod q}]\!] \quad \text{for all } i$$

Computing the characteristic vector

Task

Given $[\![v]\!]$ (modulo q), find $[\![v_0]\!], \ldots, [\![v_{q-1}]\!]$, such that:

- $v_0, \ldots, v_{q-1}$ is the C.V. of v

(Note that length of vector equals sharing modulus)

Correlated randomness

$[\![r]\!], [\![r_0]\!], \ldots, [\![r_{q-1}]\!]$

- $r \in \mathbb{Z}_q$ is random; rest is the C.V. of r

Protocol

- Compute $y \leftarrow \text{declassify}([\![v]\!] - [\![r]\!])$
- Rotate **r**, so it becomes **v**:

$$[\![v_i]\!] \leftarrow [\![r_{(i-y) \bmod q}]\!] \quad \text{for all } i$$

Moduli of sharing

$[\![v]\!]$ may be shared over a modulus different from $[\![v_0]\!], \ldots, [\![v_{q-1}]\!]$

Decomposition into digits

- Given (constants) $(r_0, r_1, \ldots, r_{n-1}) \in \mathbb{Z}^n$ be such, that $\prod_i r_i \geq 2q$
  - Denote $R_i = r_0 \cdot r_1 \cdots r_{i-1}$. (And $R_0 = 1$)
- Given $[\![v]\!]$, compute $[\![d_0]\!], \ldots, [\![d_{n-1}]\!]$, such that
  - $0 \leq d_i < r_i$
  - $\sum_{i=0}^{n-1} R_i \cdot d_i \in \{v, v+q\}$
- Actually, instead of $[\![d_i]\!]$ output its C.V. $[\![\mathbf{b}_i]\!]$

Decomposition into digits

- Given (constants) $(r_0, r_1, \ldots, r_{n-1}) \in \mathbb{Z}^n$ be such, that $\prod_i r_i \geq 2q$
  - Denote $R_i = r_0 \cdot r_1 \cdots r_{i-1}$. (And $R_0 = 1$)
- Given $[\![v]\!]$, compute $[\![d_0]\!], \ldots, [\![d_{n-1}]\!]$, such that
  - $0 \leq d_i < r_i$
  - $\sum_{i=0}^{n-1} R_i \cdot d_i \in \{v, v + q\}$
- Actually, instead of $[\![d_i]\!]$ output its C.V. $[\![\mathbf{b}_i]\!]$
- Protocol: very similar to bit decomposition
  - Correlated randomness: shares of $r \leftarrow\$ \mathbb{Z}_q$ and C.V.-s of r's digits
  - Addition circuit is similar; C.V.-s help to compute the carries

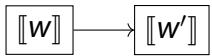
Computing high bits

Computing w^H

$$w' \leftarrow (w + \tfrac{\alpha}{2} - 1) \pmod{q}$$

$$w^H = \begin{cases} \lfloor w'/\alpha \rfloor, & w' < q - 1 \\ 0, & w' = q - 1 \end{cases}$$

Computing high bits

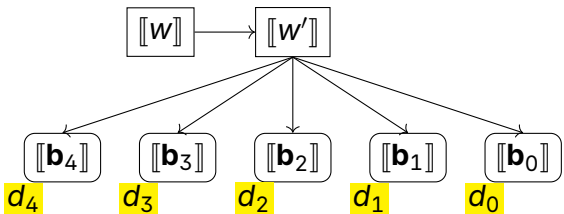


Computing w^H

$$w' \leftarrow (w + \tfrac{\alpha}{2} - 1) \pmod{q}$$

$$w^H = \begin{cases} \lfloor w'/\alpha \rfloor, & w' < q-1 \\ 0, & w' = q-1 \end{cases}$$

Computing high bits

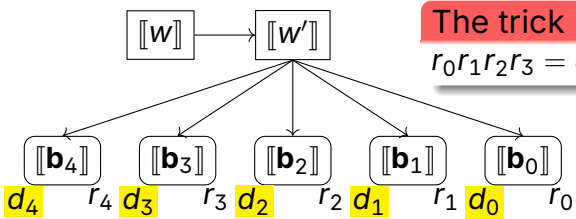


Computing w^H

$$w' \leftarrow (w + \tfrac{\alpha}{2} - 1) \pmod{q}$$

$$w^H = \begin{cases} \lfloor w'/\alpha \rfloor, & w' < q-1 \\ 0, & w' = q-1 \end{cases}$$

Computing high bits



The trick

$r_0 r_1 r_2 r_3 = \alpha$

Computing w^H

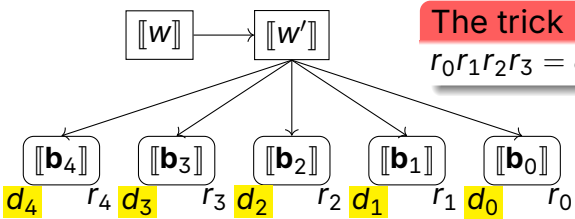
$w' \leftarrow (w + \frac{\alpha}{2} - 1) \pmod q$

$$w^H = \begin{cases} \lfloor w'/\alpha \rfloor, & w' < q-1 \\ 0, & w' = q-1 \end{cases}$$

Denote $s = (q-1)/\alpha$

Denote $D = \sum_i R_i d_i$

Computing high bits



The trick

$r_0 r_1 r_2 r_3 = \alpha$

If $d_4 < s$

(then $D = w' < q - 1$)

Computing w^H

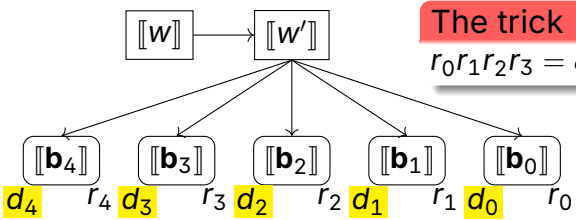
$w' \leftarrow (w + \frac{\alpha}{2} - 1) \pmod q$

$$w^H = \begin{cases} \lfloor w'/\alpha \rfloor, & w' < q - 1 \\ 0, & w' = q - 1 \end{cases}$$

Denote $s = (q-1)/\alpha$

Denote $D = \sum_i R_i d_i$

Computing high bits



The trick

$r_0 r_1 r_2 r_3 = \alpha$

Computing w^H

$w' \leftarrow (w + \frac{\alpha}{2} - 1) \pmod q$

$$w^H = \begin{cases} \lfloor w'/\alpha \rfloor, & w' < q-1 \\ 0, & w' = q-1 \end{cases}$$

Denote $s = (q-1)/\alpha$

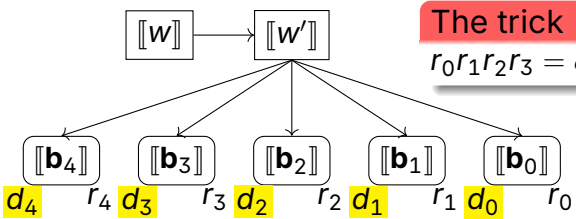
Denote $D = \sum_i R_i d_i$

If $d_4 < s$

(then $D = w' < q-1$)

return $\llbracket d_4 \rrbracket$

Computing high bits



The trick

$r_0 r_1 r_2 r_3 = \alpha$

Computing w^H

$w' \leftarrow (w + \frac{\alpha}{2} - 1) \pmod q$

$$w^H = \begin{cases} \lfloor w'/\alpha \rfloor, & w' < q-1 \\ 0, & w' = q-1 \end{cases}$$

Denote $s = (q-1)/\alpha$

Denote $D = \sum_i R_i d_i$

If $d_4 < s$

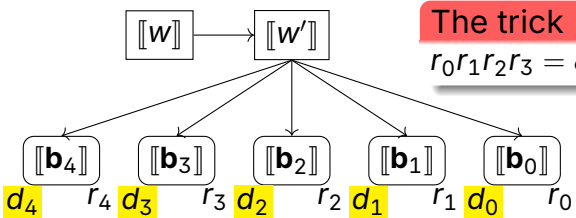
(then $D = w' < q-1$)

return $[\![d_4]\!]$

Otherwise

($D = w' + q$ (or $w' = q-1$))

Computing high bits



The trick

$r_0 r_1 r_2 r_3 = \alpha$

Computing w^H

$w' \leftarrow (w + \frac{\alpha}{2} - 1) \pmod q$

$$w^H = \begin{cases} \lfloor w'/\alpha \rfloor, & w' < q-1 \\ 0, & w' = q-1 \end{cases}$$

Denote $s = (q-1)/\alpha$

Denote $D = \sum_i R_i d_i$

If $d_4 < s$

(then $D = w' < q-1$)

return $[\![d_4]\!]$

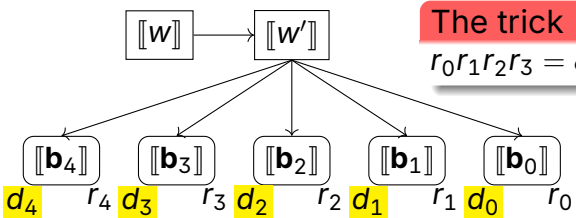
Otherwise

($D = w' + q$ (or $w' = q-1$))

return $[\![d_4]\!] - s$

(subtracts $q-1$ from D)

Computing high bits



The trick

$r_0 r_1 r_2 r_3 = \alpha$

Computing w^H

$w' \leftarrow (w + \frac{\alpha}{2} - 1) \pmod q$

$$w^H = \begin{cases} \lfloor w'/\alpha \rfloor, & w' < q-1 \\ 0, & w' = q-1 \end{cases}$$

Denote $s = (q-1)/\alpha$

Denote $D = \sum_i R_i d_i$

If $d_4 < s$

(then $D = w' < q-1$)

return $[\![d_4]\!]$

Otherwise

($D = w' + q$ (or $w' = q-1$))

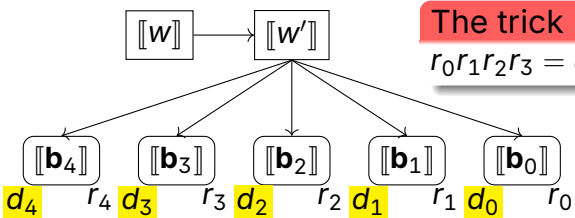
return $[\![d_4]\!] - s - [\![C]\!]$

(subtracts $q-1$ from D)

Condition C

$d_0, \ldots, d_3 = 0$

Computing high bits



The trick

$r_0 r_1 r_2 r_3 = \alpha$

Computing w^H

$w' \leftarrow (w + \frac{\alpha}{2} - 1) \pmod q$

$$w^H = \begin{cases} \lfloor w'/\alpha \rfloor, & w' < q-1 \\ 0, & w' = q-1 \end{cases}$$

Denote $s = (q-1)/\alpha$

Denote $D = \sum_i R_i d_i$

If $d_4 < s$

(then $D = w' < q-1$)

return $[\![d_4]\!]$

Otherwise

($D = w' + q$ (or $w' = q-1$))

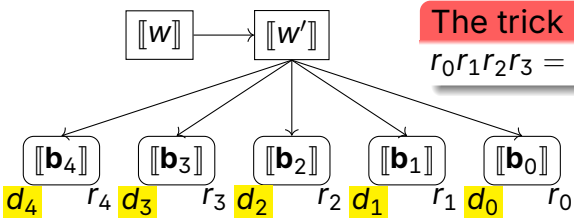
return $[\![d_4]\!] - s - [\![C]\!]$

(subtracts $q-1$ from D)

Condition C

$d_0, \ldots, d_3 = 0$, $d_4 > s$

Computing high bits



The trick

$r_0 r_1 r_2 r_3 = \alpha$

Computing w^H

$w' \leftarrow (w + \frac{\alpha}{2} - 1) \pmod q$

$$w^H = \begin{cases} \lfloor w'/\alpha \rfloor, & w' < q-1 \\ 0, & w' = q-1 \end{cases}$$

Denote $s = (q-1)/\alpha$

Denote $D = \sum_i R_i d_i$

If $d_4 < s$

(then $D = w' < q-1$)

return $[\![d_4]\!]$

Otherwise

($D = w' + q$ (or $w' = q-1$))

return $[\![d_4]\!] - s - [\![C]\!]$

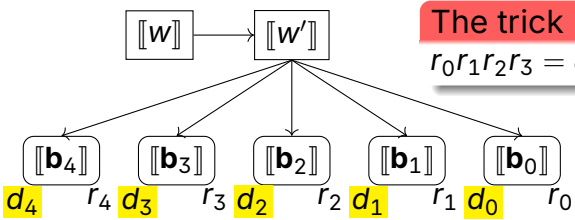
(subtracts $q-1$ from D)

Condition C

$d_0, \ldots, d_3 = 0$, $d_4 > s$

$b_{0,0}, \ldots, b_{3,0}, (\sum_{i>s} b_{4,i}) = 1$

Computing high bits



The trick

$r_0 r_1 r_2 r_3 = \alpha$

Computing w^H

$w' \leftarrow (w + \frac{\alpha}{2} - 1) \pmod q$

$$w^H = \begin{cases} \lfloor w'/\alpha \rfloor, & w' < q-1 \\ 0, & w' = q-1 \end{cases}$$

Denote $s = (q-1)/\alpha$

Denote $D = \sum_i R_i d_i$

If $d_4 < s$

(then $D = w' < q-1$)

return $\llbracket d_4 \rrbracket$

Otherwise

($D = w' + q$ (or $w' = q-1$))

return $\llbracket d_4 \rrbracket - s - \llbracket C \rrbracket$

(subtracts $q-1$ from D)

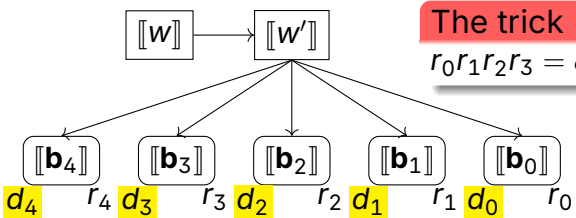
Condition C

$d_0, \ldots, d_3 = 0$, $d_4 > s$

$b_{0,0}, \ldots, b_{3,0}, (\sum_{i>s} b_{4,i}) = 1$

- Each is $\in \{0, 1\}$
- Add up, check that $= 5$

Computing high bits



The trick

$r_0 r_1 r_2 r_3 = \alpha$

The computation from $[\![\mathbf{b}_i]\!]$ is:

- Let $f(x) := \textbf{if } x < s \textbf{ then } x \textbf{ else } x - s$
- Compute $[\![C]\!]$, evaluating $\longrightarrow$ under $[\![\cdot]\!]$
- **return** $[\![f(d_4)]\!] - [\![C]\!]$

Computing w^H

$w' \leftarrow (w + \frac{\alpha}{2} - 1) \pmod q$

$$w^H = \begin{cases} \lfloor w'/\alpha \rfloor, & w' < q-1 \\ 0, & w' = q-1 \end{cases}$$

Denote $s = (q-1)/\alpha$

Denote $D = \sum_i R_i d_i$

Condition C

$d_0, \ldots, d_3 = 0$, $d_4 > s$

$b_{0,0}, \ldots, b_{3,0}, (\sum_{i>s} b_{4,i}) = 1$

- Each is $\in \{0, 1\}$
- Add up, check that $= 5$

Limited range zero check

Task

- Given $[\![v]\!]$ (modulo q) with $v \leq B$
- Compute $[\![b]\!]$, $b = [v = 0]$

Limited range zero check

Task

- Given $[\![v]\!]$ (modulo q) with $v \leq B$
- Compute $[\![b]\!]$, $b = [v = 0]$

Denote $a = \lfloor q/B \rfloor$

Correlated randomness

- $[\![r]\!]$, $[\![\mathbf{t}]\!]$, $r \leftarrow\!\$\ \mathbb{Z}_q$
- $\mathbf{t}$ is C.V. of $\lfloor r/a \rfloor$
  - …of length B

Limited range zero check

Task

- Given $[\![v]\!]$ (modulo q) with $v \leq B$
- Compute $[\![b]\!]$, $b = [v = 0]$

Denote $a = \lfloor q/B \rfloor$

Correlated randomness

- $[\![r]\!]$, $[\![\mathbf{t}]\!]$, $r \leftarrow\!\!\$\ \mathbb{Z}_q$
- $\mathbf{t}$ is C.V. of $\lfloor r/a \rfloor$
  - …of length B

Protocol

- $y \leftarrow \text{declassify}([\![r]\!] + a \cdot [\![v]\!])$
- Output $[\![t_{\lfloor y/a \rfloor}]\!]$

Limited range zero check

Task

- Given $[\![v]\!]$ (modulo q) with $v \leq B$
- Compute $[\![b]\!]$, $b = [v = 0]$

Denote $a = \lfloor q/B \rfloor$

Correlated randomness

- $[\![r]\!]$, $[\![\mathbf{t}]\!]$, $r \leftarrow\$ \mathbb{Z}_q$
- $\mathbf{t}$ is C.V. of $\lfloor r/a \rfloor$
  - ...of length B

Protocol

- $y \leftarrow \text{declassify}([\![r]\!] + a \cdot [\![v]\!])$
- Output $[\![t_{\lfloor y/a \rfloor}]\!]$

Correctness

Must get $\lfloor \frac{r}{a} \rfloor = \lfloor \frac{y}{a} \rfloor$ iff $v = 0$

- $\lfloor \frac{y}{a} \rfloor = \lfloor \frac{r+av}{a} \rfloor = \lfloor \frac{r}{a} \rfloor + v$
  - But if $r + av \geq q$ then $y \leq r - a$ and $\lfloor \frac{y}{a} \rfloor \leq \lfloor \frac{r}{a} \rfloor - 1$

Computing whether $[\![w]\!] < c$

- Say that $[\![w]\!]$ overflows, if $[\![w]\!]_0 + [\![w]\!]_1 \geq q$
  - Let there be a protocol $OF([\![w]\!]) \mapsto [\![b]\!]$
- Let c be shared as $[\![c]\!]_0 = 0$, $[\![c]\!]_1 = c$
- Let
  - $[\![X]\!] = OF([\![w]\!])$
  - $[\![Y]\!] = OF([\![w]\!] - [\![c]\!])$
  - $[\![Z]\!]_0 = 0$, $[\![Z]\!]_1 \in \{0, 1\}$, $[\![Z]\!]_1 = 1$ iff $[\![w]\!]_1 \geq c$

Computing whether $[\![w]\!] < c$

- Say that $[\![w]\!]$ overflows, if $[\![w]\!]_0 + [\![w]\!]_1 \geq q$
  - Let there be a protocol $OF([\![w]\!]) \mapsto [\![b]\!]$
- Let c be shared as $[\![c]\!]_0 = 0$, $[\![c]\!]_1 = c$
- Let
  - $[\![X]\!] = OF([\![w]\!])$
  - $[\![Y]\!] = OF([\![w]\!] - [\![c]\!])$
  - $[\![Z]\!]_0 = 0$, $[\![Z]\!]_1 \in \{0, 1\}$, $[\![Z]\!]_1 = 1$ iff $[\![w]\!]_1 \geq c$

X	Y	Z	R
0	0	0	
0	0	1	
0	1	0	
0	1	1	
1	0	0	
1	0	1	
1	1	0	
1	1	1	

Computing whether $[\![w]\!] <$

First row in table

- $0 \leq [\![w]\!]_0 + [\![w]\!]_1 < q$
- $0 \leq [\![w]\!]_0 + ([\![w]\!]_1 - c) \bmod q < q$
- $0 \leq [\![w]\!]_1 < c$

- Say that $[\![w]\!]$ overflows, if $[\![w]\!]_0 + [\![w]\!]_1 \geq q$
  - Let there be a protocol $OF([\![w]\!]) \mapsto [\![b]\!]$
- Let c be shared as $[\![c]\!]_0 = 0$, $[\![c]\!]_1 = c$
- Let
  - $[\![X]\!] = OF([\![w]\!])$
  - $[\![Y]\!] = OF([\![w]\!] - [\![c]\!])$
  - $[\![Z]\!]_0 = 0$, $[\![Z]\!]_1 \in \{0, 1\}$, $[\![Z]\!]_1 = 1$ iff $[\![w]\!]_1 \geq c$

X	Y	Z	R
0	0	0	
0	0	1	
0	1	0	
0	1	1	
1	0	0	
1	0	1	
1	1	0	
1	1	1	

Computing whether $[\![w]\!] <$

- Say that $[\![w]\!]$ overflows, if $[\![w]\!]_0 + [\![w]\!]_1 \geq$
  - Let there be a protocol $OF([\![w]\!]) \mapsto [\![b]\!]$
- Let c be shared as $[\![c]\!]_0 = 0$, $[\![c]\!]_1 = c$
- Let
  - $[\![X]\!] = OF([\![w]\!])$
  - $[\![Y]\!] = OF([\![w]\!] - [\![c]\!])$
  - $[\![Z]\!]_0 = 0$, $[\![Z]\!]_1 \in \{0, 1\}$, $[\![Z]\!]_1 = 1$ iff $[\![w]\!]_1 \geq c$

First row in table

- $0 \leq [\![w]\!]_0 + [\![w]\!]_1 < q$
- $0 \leq [\![w]\!]_0 + ([\![w]\!]_1 - c) \bmod q < q$
- $0 \leq [\![w]\!]_1 < c$
- $0 \leq [\![w]\!]_0 + [\![w]\!]_1 + q - c < q$
- $c - q \leq [\![w]\!]_0 + [\![w]\!]_1 < c$

0	0	1
0	1	0
0	1	1
1	0	0
1	0	1
1	1	0
1	1	1

Computing whether $[\![w]\!] < c$

- Say that $[\![w]\!]$ overflows, if $[\![w]\!]_0 + [\![w]\!]_1 \geq q$
  - Let there be a protocol $OF([\![w]\!]) \mapsto [\![b]\!]$
- Let c be shared as $[\![c]\!]_0 = 0$, $[\![c]\!]_1 = c$
- Let
  - $[\![X]\!] = OF([\![w]\!])$
  - $[\![Y]\!] = OF([\![w]\!] - [\![c]\!])$
  - $[\![Z]\!]_0 = 0$, $[\![Z]\!]_1 \in \{0, 1\}$, $[\![Z]\!]_1 = 1$ iff $[\![w]\!]_1 \geq c$

X	Y	Z	R
0	0	0	0
0	0	1	
0	1	0	
0	1	1	
1	0	0	
1	0	1	
1	1	0	
1	1	1	

Computing whether $[\![w]\!] < c$

- Say that $[\![w]\!]$ overflows, if $[\![w]\!]_0 + [\![w]\!]_1 \geq q$
  - Let there be a protocol $OF([\![w]\!]) \mapsto [\![b]\!]$
- Let c be shared as $[\![c]\!]_0 = 0$, $[\![c]\!]_1 = c$
- Let
  - $[\![X]\!] = OF([\![w]\!])$
  - $[\![Y]\!] = OF([\![w]\!] - [\![c]\!])$
  - $[\![Z]\!]_0 = 0$, $[\![Z]\!]_1 \in \{0, 1\}$, $[\![Z]\!]_1 = 1$ iff $[\![w]\!]_1 \geq c$

X	Y	Z	R
0	0	0	0
0	0	1	1
0	1	0	1
0	1	1	$\perp$
1	0	0	$\perp$
1	0	1	0
1	1	0	0
1	1	1	1

Computing whether $[\![w]\!] < c$

- Say that $[\![w]\!]$ overflows, if $[\![w]\!]_0 + [\![w]\!]_1 \geq q$
  - Let there be a protocol $OF([\![w]\!]) \mapsto [\![b]\!]$
- Let c be shared as $[\![c]\!]_0 = 0$, $[\![c]\!]_1 = c$
- Let
  - $[\![X]\!] = OF([\![w]\!])$
  - $[\![Y]\!] = OF([\![w]\!] - [\![c]\!])$
  - $[\![Z]\!]_0 = 0$, $[\![Z]\!]_1 \in \{0, 1\}$, $[\![Z]\!]_1 = 1$ iff $[\![w]\!]_1 \geq c$
- Output $[\![R]\!] = [\![Y]\!] - [\![X]\!] + [\![Z]\!]$

X	Y	Z	R
0	0	0	0
0	0	1	1
0	1	0	1
0	1	1	$\perp$
1	0	0	$\perp$
1	0	1	0
1	1	0	0
1	1	1	1

Computing overflow of $[\![w]\!]$ (small sharing modulus r)

- Let $[\![u]\!]_0 \leftarrow [\![w]\!]_0$ and $[\![u]\!]_1 \leftarrow [\![v]\!]_1$
- Think of $[\![u]\!]$ being shared modulo q, where $q \geq 2r - 1$
- Let $[\![\mathbf{v}]\!]$ be the C.V. of $[\![u]\!]$
- Compute $[\![c]\!] \leftarrow \sum_{i=r}^{q-1} [\![v_i]\!]$
- (The next algorithm also needs $[\![b]\!] \leftarrow [\![v_{r-1}]\!]$)

Computing overflow of $[\![w]\!]$

- Fix a radix basis $(r_0, \ldots, r_{n-1})$, s.t. $R_n > q$
- Also fix moduli $q_0, \ldots, q_{n-1}$, s.t. $q_i \geq 2r_i - 1$
- Party 0 splits $[\![w]\!]_0$ to digits $[\![v_0]\!]_0, \ldots, [\![v_{n-1}]\!]_0$
- Party 1 splits $[\![w]\!]_1 + (R_n - q)$ to digits $[\![v_0]\!]_1, \ldots, [\![v_{n-1}]\!]_1$
- Compute overflow bits $[\![b_i]\!]$, $[\![c_i]\!]$ for $[\![v_i]\!]$. Result is

$$c_{n-1} \vee (c_{n-2} \wedge b_{n-1}) \vee (c_{n-3} \wedge b_{n-2} \wedge b_{n-1}) \vee \cdots \vee (c_0 \wedge b_1 \wedge \cdots \wedge b_{n-1})$$

- This can be privately computed as follows…

Computing that boolean formula

- $c_{n-1} \vee (c_{n-2} \wedge b_{n-1}) \vee (c_{n-3} \wedge b_{n-2} \wedge b_{n-1}) \vee \cdots \vee (c_0 \wedge b_1 \wedge \cdots \wedge b_{n-1})$
  - Note that $b_i \wedge c_i = \text{false}$ for all i
  - Hence at most one disjunct is true
- $[\![b_i]\!]$, $[\![c_i]\!]$ are additively shared. We do not have "$\wedge$" and "$\vee$".
- Define *weights*: $\gamma_i \leftarrow 2^i$. $\beta_0 \leftarrow 0$. $\beta_i \leftarrow 2^{i-1}$
- Compute $[\![S]\!] \leftarrow \sum_{i=0}^{n-1} \gamma_i \cdot [\![c_i]\!] + \sum_{i=0}^{n-1} \beta_i \cdot [\![b_i]\!]$
  - Let the sharing modulus be $\geq 2^n$. Then the computation does not overflow
- $[\![S]\!] \geq 2^{n-1}$ iff the boolean formula is true
- Use the C.V. of $[\![S]\!]$ to find out

Security against active adversaries

- How does one party verify that the other party is following the protocol?
- “Active security”: a deviation will be caught
  - A wrong answer will be impossible
  - leaks due to deviation cannot happen
- “Active privacy”: Nothing leaks to the deviating party
  - . . . until the result has been opened
    - the amount of “wrongness” in the result may give information to the malicious party
- The (sub-)protocols we’ve seen so far only provide security against passive adversaries

BeDOZa-style MACs

- Let the sharing be over a *field* $\mathbb{Z}_q$
- P_i has a private *MAC key* $\Delta_i \in \mathbb{Z}_q$
- The sharing $\langle\!\langle x \rangle\!\rangle$ of $x \in \mathbb{Z}_q$ consists of
  - $[\![x]\!]$ (as in passive case)
  - $[\![[\![x]\!]_0 \cdot \Delta_1]\!]$ and $[\![[\![x]\!]_1 \cdot \Delta_0]\!]$ (the "MAC"-s)
- Whenever P_i sends some $[\![y]\!]_i$ to P_j, it also sends $[\![[\![y]\!]_i \cdot \Delta_j]\!]_i$

BeDOZa-style MACs

- Let the sharing be over a *field* $\mathbb{Z}_q$
- P_i has a private *MAC key* $\Delta_i \in \mathbb{Z}_q$
- The sharing $\langle\!\langle x \rangle\!\rangle$ of $x \in \mathbb{Z}_q$ consists of
  - $[\![x]\!]$ (as in passive case)
  - $[\![[\![x]\!]_0 \cdot \Delta_1]\!]$ and $[\![[\![x]\!]_1 \cdot \Delta_0]\!]$ (the "MAC"-s)
- Whenever P_i sends some $[\![y]\!]_i$ to P_j, it also sends $[\![[\![y]\!]_i \cdot \Delta_j]\!]_i$
- Suppose P_i wants to send some $[\![y]\!]'_i$ instead. P_i knows $d = [\![y]\!]'_i - [\![y]\!]_i$
- In order to pass, P_i has to change the MAC share by $\Delta_j \cdot d$
- If P_i is successful, then Δ_j can be obtained from his knowledge

BeDOZa style MACs (cont.)

- Cheating against P_i is at least as hard as guessing a random element of $\mathbb{Z}_q$
- How hard is this? Depends on q
- If q is small, then we just use more Δ-s
  - Instead of Δ_i, have $\Delta_{i,1}, \ldots, \Delta_{i,k}$, s.t. q^k is sufficiently large
- If several different q-s are used in the protocol, then have different Δ-s for each of them

MACs in computations

- MACs are linear. Linear operations are still local
- Correlated randomness includes MACs
- If some P_i needs to enter a new value v into the computation:
  - Correlated randomness includes a random $\langle\langle r \rangle\rangle$ (with all the MACs)
  - Open r to P_i
  - P_i sends $\delta = v - r$ to everybody
  - Take $\langle\langle v \rangle\rangle \leftarrow \langle\langle r \rangle\rangle + \delta$

Communication complexity of MACs

- MAC shares are included in correlated randomness. This gets bigger
- MAC shares are communicated in the online phase
  - In general, many values are sent in each round. Each has accompanying MAC share(s)
- But the receiver already knows, what he is supposed to receive
- Do not send the shares themselves. Send their hash
- I.e. The online communication complexity is minimal

MACs for multiplications and high bits

- Correlated randomness also gets MACs
- The operations in the title, together with correlated randomness, are “linear” operations
- Other “linear” operations:
  - Characteristic vector
  - equality check
  - Generation of random bits

MACs for less-than

- Operation: $[\![w]\!]_i \mapsto ([\![v_0]\!]_i, [\![v_1]\!]_i, \ldots, [\![v_{n-1}]\!]_i)$
- $[\![w]\!]_i$ has MAC for party P_{1-i}. How do we get MACs on $[\![v_j]\!]_i$?

MACs for less-than

- Operation: $[\![w]\!]_i \mapsto ([\![v_0]\!]_i, [\![v_1]\!]_i, \ldots, [\![v_{n-1}]\!]_i)$
- $[\![w]\!]_i$ has MAC for party P_{1-i}. How do we get MACs on $[\![v_j]\!]_i$?

- Consider the following correlated randomness:
  - $s \in \mathbb{Z}_q$, known to party 0 only
  - $[\![\Delta_1 \cdot s]\!]$ (i.e. MAC on s)
  - $\langle\!\langle d_0 \rangle\!\rangle, \ldots, \langle\!\langle d_{n-1} \rangle\!\rangle$, for digits $d_0, \ldots, d_{n-1}$ of s

MACs for less-than

- Operation: $[\![w]\!]_i \mapsto ([\![v_0]\!]_i, [\![v_1]\!]_i, \ldots, [\![v_{n-1}]\!]_i)$
- $[\![w]\!]_i$ has MAC for party P_{1-i}. How do we get MACs on $[\![v_j]\!]_i$?

- Consider the following correlated randomness:
  - $s \in \mathbb{Z}_q$, known to party 0 only
  - $[\![\Delta_1 \cdot s]\!]$ (i.e. MAC on s)
  - $\langle\!\langle d_0 \rangle\!\rangle, \ldots, \langle\!\langle d_{n-1} \rangle\!\rangle$, for digits $d_0, \ldots, d_{n-1}$ of s

- P_0 sends $t = [\![w]\!]_0 - s$ to P_1
  - With MAC check
- P_0 updates $[\![w]\!]_0 := s$
- P_1 updates $[\![w]\!]_1 := [\![w]\!]_1 + t$

MACs for less-than

- Operation: $[\![w]\!]_i \mapsto ([\![v_0]\!]_i, [\![v_1]\!]_i, \ldots, [\![v_{n-1}]\!]_i)$
- $[\![w]\!]_i$ has MAC for party P_{1-i}. How do we get MACs on $[\![v_j]\!]_i$?

- Consider the following correlated randomness:
  - $s \in \mathbb{Z}_q$, known to party 0 only
  - $[\![\Delta_1 \cdot s]\!]$ (i.e. MAC on s)
  - $\langle\!\langle d_0 \rangle\!\rangle, \ldots, \langle\!\langle d_{n-1} \rangle\!\rangle$, for digits $d_0, \ldots, d_{n-1}$ of s
- P_0 sends $t = [\![w]\!]_0 - s$ to P_1
  - With MAC check
- P_0 updates $[\![w]\!]_0 := s$
- P_1 updates $[\![w]\!]_1 := [\![w]\!]_1 + t$
- Now we have MACs on P_0's shares of digits of $[\![w]\!]$
- Same MACs work for P_0's shares of digits of $[\![w]\!] - [\![c]\!]$
- Other details of "less than" also support MACs on P_0's shares

Active security against one party only

- Rejection check is actively secure against phone, passively secure against server
  - It is actively *private* against server
- If checks pass, then the signature will be revealed to phone only
- Phone verifies the signature. If bad, then server has cheated

Active security against one party only

- Rejection check is actively secure against phone, passively secure against server
  - It is actively *private* against server
- If checks pass, then the signature will be revealed to phone only
- Phone verifies the signature. If bad, then server has cheated
- Server's best strategy to learn something: follow the protocol

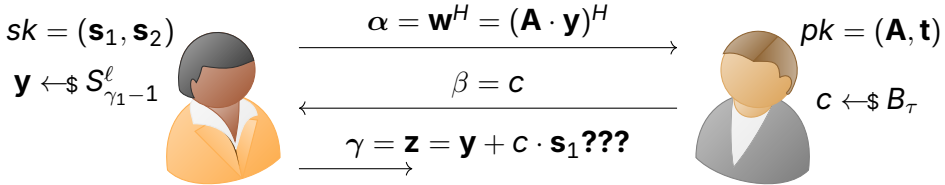
Source of correlated randomness?

- This randomness could be generated ahead-of-time
  - There are protocols for certain kinds of correlated randomness
  - These protocols are (more) expensive

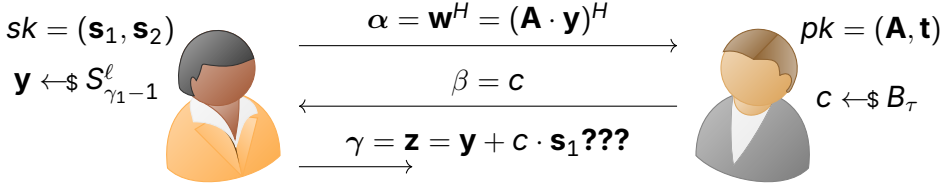
Source of correlated randomness?

- This randomness could be generated ahead-of-time
  - There are protocols for certain kinds of correlated randomness
  - These protocols are (more) expensive
- We just add an extra party
  - Randomness going to one of the parties (Phone) can be expanded from a single random seed
  - We keep that extra party close to the other party (Server)
    - But under independent control
- In a 3-party protocol (P and S and CRP), we can handle one malicious party

Making $\mathbf{w}^H$ public? "Dilithium identification protocol"



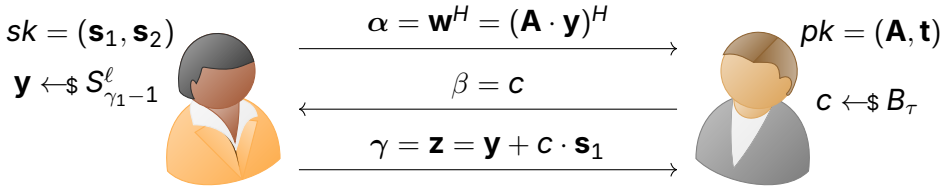
Making $\mathbf{w}^H$ public? "Dilithium identification protocol"



Alice checks:

- Would the value of $\mathbf{y} + c \cdot \mathbf{s}_1$ be achievable for all possible $\tilde{\mathbf{s}}_1 \in R_1^n$?
  - Is it the case that $\forall \tilde{\mathbf{s}}_1 \in S^{\ell}_{\eta}\ \exists \tilde{\mathbf{y}} \in S^{\ell}_{\gamma_1 - 1}$, s.t. $\mathbf{y} + \beta \cdot \mathbf{s}_1 = \tilde{\mathbf{y}} + c \cdot \tilde{\mathbf{s}}_1$?
- Would the value of $\mathbf{w}^L - c \cdot \mathbf{s}_2$ be achievable for all possible $\tilde{\mathbf{s}}_2 \in R_1^m$?
- I.e. aren't these values "dangerously off to one side"?

Making $\mathbf{w}^H$ public? "Dilithium identification protocol"

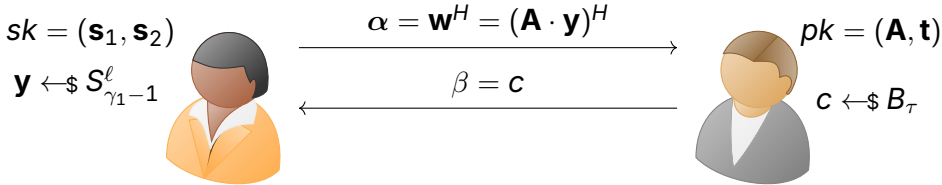


Alice checks:

- Would the value of $\mathbf{y} + c \cdot \mathbf{s}_1$ be achievable for all possible $\tilde{\mathbf{s}}_1 \in R_1^n$?
  - Is it the case that $\forall \tilde{\mathbf{s}}_1 \in S^{\ell}_{\eta}\ \exists \tilde{\mathbf{y}} \in S^{\ell}_{\gamma_1 - 1}$, s.t. $\mathbf{y} + \beta \cdot \mathbf{s}_1 = \tilde{\mathbf{y}} + c \cdot \tilde{\mathbf{s}}_1$?
- Would the value of $\mathbf{w}^L - c \cdot \mathbf{s}_2$ be achievable for all possible $\tilde{\mathbf{s}}_2 \in R_1^m$?
- I.e. aren't these values "dangerously off to one side"?

YES

Making $\mathbf{w}^H$ public? "Dilithium identification protocol"



Alice checks:

- Would the value of $\mathbf{y} + c \cdot \mathbf{s}_1$ be achievable for all possible $\tilde{\mathbf{s}}_1 \in R_1^n$?
  - Is it the case that $\forall \tilde{\mathbf{s}}_1 \in S^{\ell}_{\eta}\ \exists \tilde{\mathbf{y}} \in S^{\ell}_{\gamma_1 - 1}$, s.t. $\mathbf{y} + \beta \cdot \mathbf{s}_1 = \tilde{\mathbf{y}} + c \cdot \tilde{\mathbf{s}}_1$?
- Would the value of $\mathbf{w}^L - c \cdot \mathbf{s}_2$ be achievable for all possible $\tilde{\mathbf{s}}_2 \in R_1^m$?
- I.e. aren't these values "dangerously off to one side"?

NO

Making $\mathbf{w}^H$ public? "Dilithium identification protocol"

$sk = (\mathbf{s}_1, \mathbf{s}_2)$

$\mathbf{y} \leftarrow\$ S^{\ell}_{\gamma_1 - 1}$

no-abort Honest Verifier Zero-Knowledge

Bob forgets that this interaction took place at all

$pk = (\mathbf{A}, \mathbf{t})$

$c \leftarrow\$ B_{\tau}$

Alice checks:

- Would the value of $\mathbf{y} + c \cdot \mathbf{s}_1$ be achievable for all possible $\tilde{\mathbf{s}}_1 \in R_1^n$?
  - Is it the case that $\forall \tilde{\mathbf{s}}_1 \in S^{\ell}_{\eta}\ \exists \tilde{\mathbf{y}} \in S^{\ell}_{\gamma_1 - 1}$, s.t. $\mathbf{y} + \beta \cdot \mathbf{s}_1 = \tilde{\mathbf{y}} + c \cdot \tilde{\mathbf{s}}_1$?
- Would the value of $\mathbf{w}^L - c \cdot \mathbf{s}_2$ be achievable for all possible $\tilde{\mathbf{s}}_2 \in R_1^m$?
- I.e. aren't these values "dangerously off to one side"?

NO

Discussion

- Cozzo & Smart [IMACC 2019]: should compute $H(\mu, \mathbf{w}^H)$ under MPC
- Barthe et al. [EC 2018]: introduce a hardness assumption to open $\mathbf{w}^H$
- Problem occurs also in hardware security
  - Migliore et al. [ACNS 2019] ignore it
  - Coron et al. [TCHES 2023(4)] [TCHES 2024(4)] and Azouaoui et al. [TCHES 2023(4)] recognize it
    - "Future work", "need a hardness assumption", "conjecture that not a problem"
- Bienstock et al. [ePrint 2025/1163] modify Dilithium, such that security of opening $\mathbf{w}^H$ follows from MLWE assumption

Discussion

- Cozzo & Smart [IMACC 2019]: should compute $H(\mu, \mathbf{w}^H)$ under MPC
- Barthe et al. [EC 2018]: introduce a hardness assumption to open $\mathbf{w}^H$
- Problem occurs also in hardware security
  - Migliore et al. [ACNS 2019] ignore it
  - Coron et al. [TCHES 2023(4)] [TCHES 2024(4)] and Azouaoui et al. [TCHES 2023(4)] recognize it
    - "Future work", "need a hardness assumption", "conjecture that not a problem"
- Bienstock et al. [ePrint 2025/1163] modify Dilithium, such that security of opening $\mathbf{w}^H$ follows from MLWE assumption
- We modify Bienstock et al.'s argument, use MLWR on unmodified Dilithium
  - Hopefully the argument can be improved. See [ePrint 2025/675]

Discussion

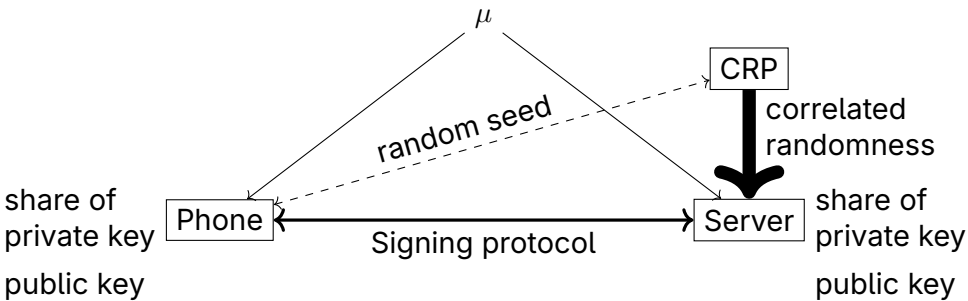
- Cozzo & Smart [IMACC 2019]: should compute $H(\mu, \mathbf{w}^H)$ under MPC

Azouaoui et al. [TCHES 2023(4)]

When the boundary checks do not pass, the reduction from LWR to LWE does not apply immediately since the distribution on $\mathbf{w}_1$ changes slightly. Leaving $\mathbf{w}_1$ unmasked therefore requires the additional explicit assumption that the corresponding LWR problem is hard. Since the number of rounded bits is significantly higher than the error that is added in the LWE problem, we conjecture that $\mathbf{w}_1$ does not require side-channel countermeasures. The same expectation was also shared by Vadim Lyubashevky in a personal communication and publicly during RWPQC23. To be conservative, we nevertheless study the option

- We modify Bienstock et al.'s argument, use MLWR on unmodified Dilithium
  - Hopefully the argument can be improved. See [ePrint 2025/675]

Benchmarking set-up



Network traffic

Key generation: traffic volume

ML-DSA-	P↔S	CRP→S
-44	920.52 KiB	1.42 MiB
-65	1760.22 KiB	2.66 MiB
-87	3704.24 KiB	4.44 MiB

Signing attempt: traffic volume

ML-DSA-	P↔S	CRP→S
-44	186.72 KiB	55.05 MiB
-65	244.86 KiB	71.23 MiB
-87	328.50 KiB	95.95 MiB

Network traffic

Key generation: traffic volume

ML-DSA-	P↔S	CRP→S
-44	920.52 KiB	1.42 MiB
-65	1760.22 KiB	2.66 MiB
-87	3704.24 KiB	4.44 MiB

Signing attempt: traffic volume

ML-DSA-	P↔S	CRP→S
-44	186.72 KiB	55.05 MiB
-65	244.86 KiB	71.23 MiB
-87	328.50 KiB	95.95 MiB

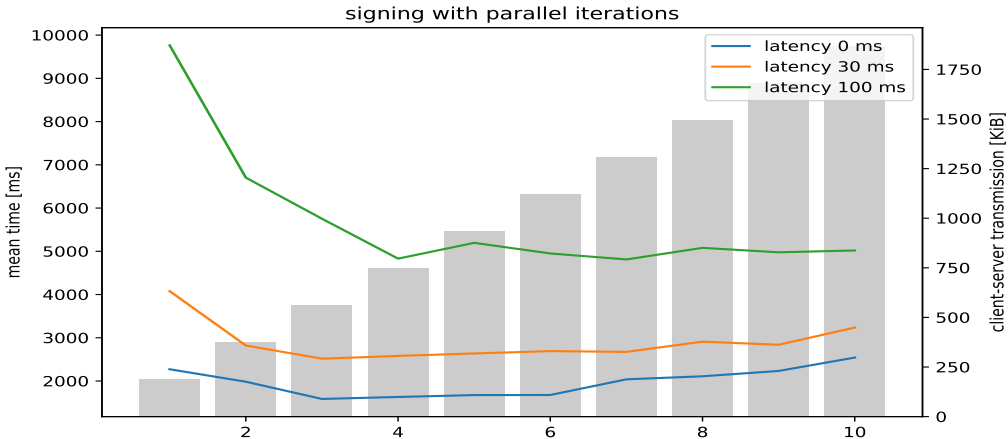
Key generation: num. of rounds

3

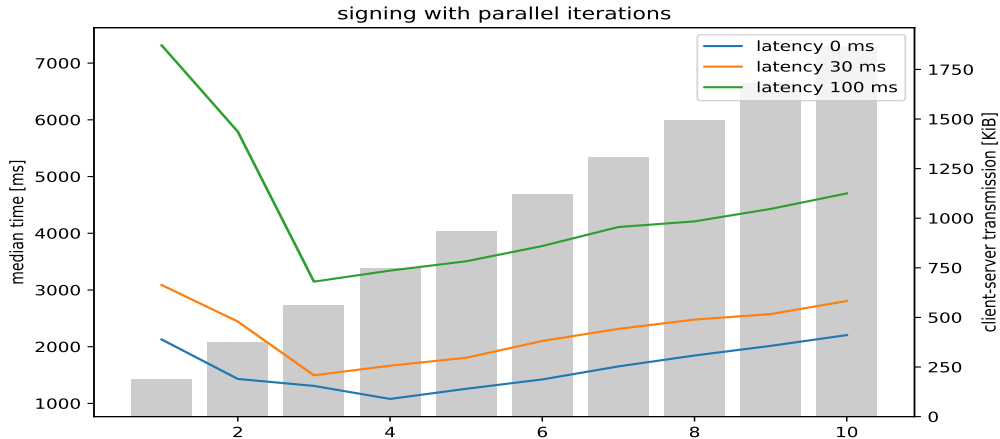
Signing attempt: num. of rounds

14

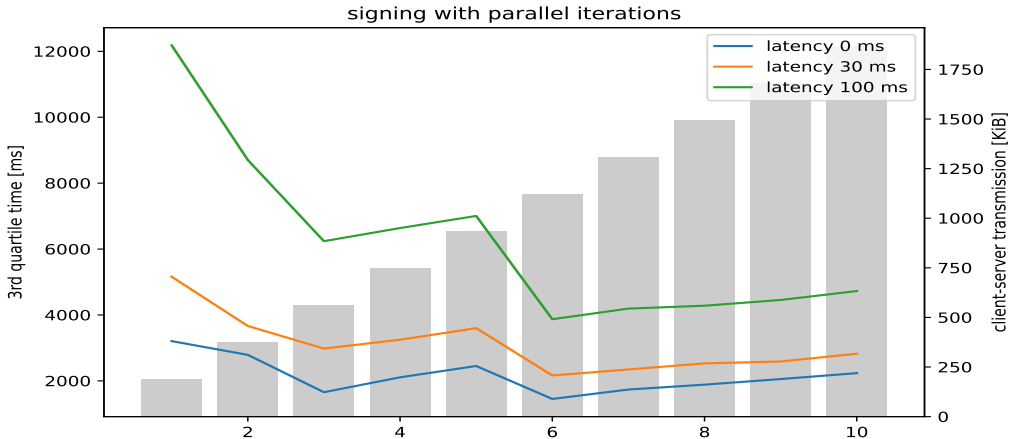
Time to create a signature (mean)



Time to create a signature (median)



Time to create a signature (3rd quartile)



Conclusion

- We obtain our desiderata (two-party, interchangable with a standardized scheme, passes known answer tests)
- The online complexity is good enough
- The setting with an extra CRP may be challenging wrt. certification
  - But hopefully can be set up and presented in an approvable manner
- The amount of correlated randomness is a bit too high
  - But probably not show-stoppingly high