Parameter set

Matrix equation sizes 2×4 and 4×1 ($n = 2$, $m = 4$, $k = 1$); bound $B = 4$; plaintext modulus $q = 8191$; Z_p prime p is the one used for scalar operations in SECP256k1 ($2^{256} - 2^{32} - 977$); modulus size 1024.

Measurements, SECP256k1

Network transfer

Transfer	size
Shared randomness (to verifier)	6.13 Mb
Random challenge (to prover)	3.05 Mb
Folding info (to verifier)	64 b × 17
Folding info (to prover)	32 b × 17
Inner product info (to verifier)	64 b
Inner product info (to prover)	32 b
Inner product info (to verifier)	96 b

Single thread, 6-Core Intel Core i7 @ 2.6 GHz

main	inner product	folding	time	main	inner product	folding	time
P1			1.27s				
				V1			46.3ms
P2			30.3s	V2			30.1s
					V1		136µs
	P1		9.11ms		V2		7.67ms
		P1	21.7s				
						V1	96.2µs
		P2	16.8s			V2	16.7s
	P2		765µs				
					V3		2.68µs
	P3		127µs				
					V4 (check happens here)		993µs
Prover time			$\sim$ 70 s	Verifier time			$\sim$ 47 s

6 threads, 6-Core Intel Core i7 @ 2.6 GHz

main	inner product	folding	time	main	inner product	folding	time
P1			790ms				
				V1			35.9ms
P2			6.01s	V2			6.58s
					V1		171µs
	P1		23.0ms		V2		17.5ms
		P1	5.23s				
						V1	98.6µs
		P2	4.42s			V2	4.28s
	P2		931µs				
					V3		4.88µs
	P3		145µs				
					V4 (check happens here)		1.23ms
Prover time			$\sim$ 15.7 s	Verifier time			$\sim$ 10.9 s

Totals

	i7 @ 2.6 GHz, 1 thread	i7 @ 2.6 GHz, 6 threads
Prover time	70s	15.7s
Verifier time	47s	10.9s
Prover transfers	6 Mb	
Verifier transfers	3 Mb	

Measurements, Curve25519

Note: not all optimized batch multiplications algorithms are in use here. The numbers can be improved.

Network transfer

Transfer	size
Shared randomness (to verifier)	30.6 Mb
Random challenge (to prover)	3.05 Mb
Folding info (to verifier)	320 b × 17
Folding info (to prover)	32 b × 17
Inner product info (to verifier)	320 b
Inner product info (to prover)	32 b
Inner product info (to verifier)	96 b

Single thread, 6-Core Intel Core i7 @ 2.6 GHz

main	inner product	folding	time	main	inner product	folding	time
P1			4.43s				
				V1			685ms
P2			23.8s	V2			22.5s
					V1		60.3µs
	P1		23.3ms		V2		24.5ms
		P1	12.1s				
						V1	239µs
		P2	20.8s			V2	18.4s
	P2		351µs				
					V3		6.46µs
	P3		134µs				
					V4 (check happens here)		506µs
Prover time			$\sim 61\,\mathrm{s}$	Verifier time			$\sim 41\,\mathrm{s}$

6 threads, 6-Core Intel Core i7 @ 2.6 GHz

main	inner product	folding	time	main	inner product	folding	time
P1			4.65ms				
				V1			603ms
P2			8.91s	V2			7.84s
					V1		62.6µs
	P1		39.2ms		V2		30.8ms
		P1	4.09s				
						V1	250µs
		P2	5.76s			V2	4.29s
	P2		405µs				
					V3		9.47µs
	P3		151µs				
					V4 (check happens here)		577µs
Prover time			$\sim 18.8\,\mathrm{s}$	Verifier time			$\sim 12.1\,\mathrm{s}$

Totals

	i7 @ 2.6 GHz, 1 thread	i7 @ 2.6 GHz, 6 threads
Prover time	61s	18.8s
Verifier time	41s	12.1s
Prover transfers	30.6 Mb	
Verifier transfers	3 Mb	

Notes

- The initial transfers consist mostly of random numbers. Their size can be reduced significantly (to the order of several bytes) if one can just transfer a random seed and trust the other party to generate the randoms.

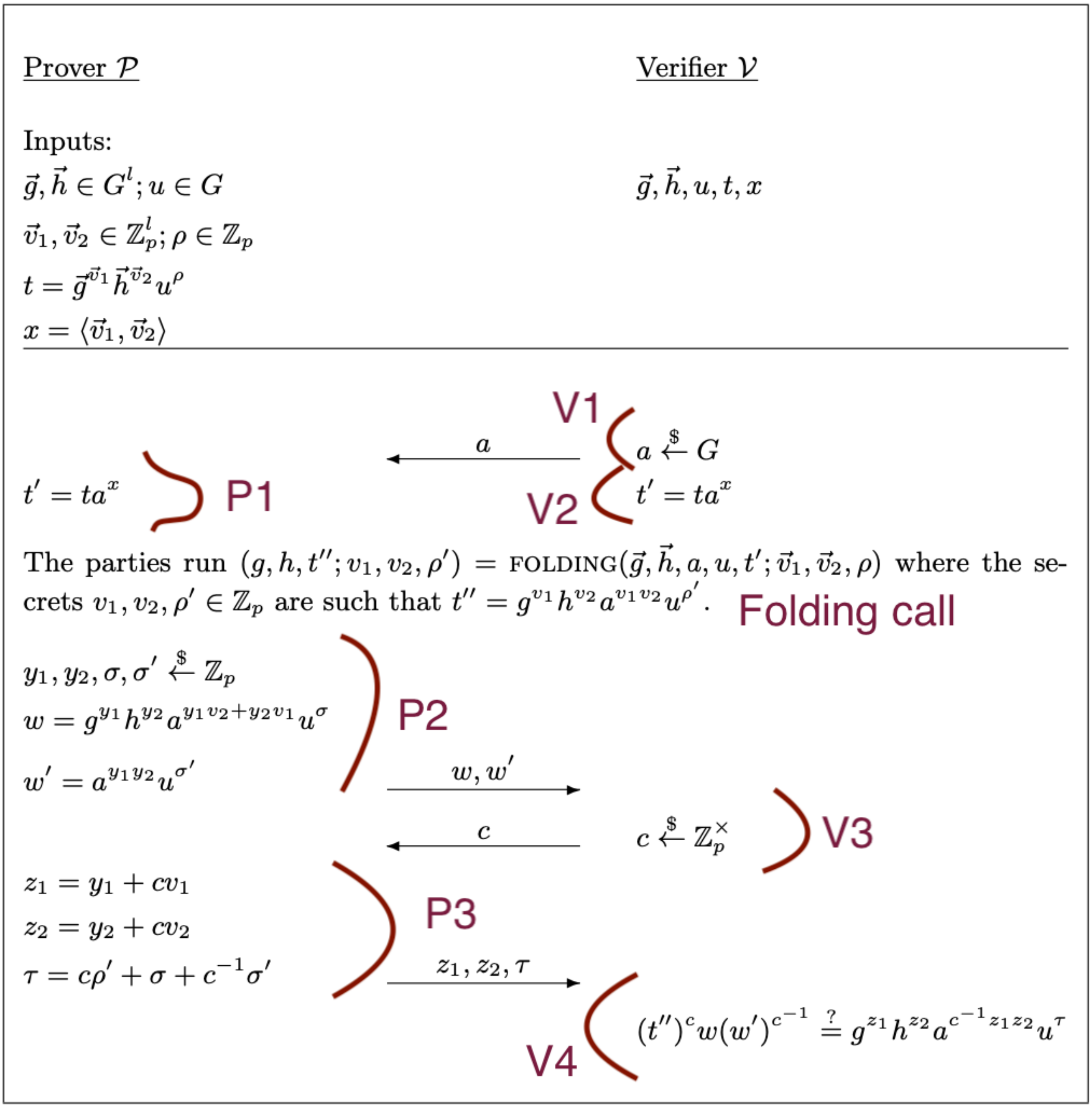
- There are many consecutive transfers during folding stages. It may be possible to pack them into a single transfer from each size (the verifier prepares an array of randoms c, the prover calculates t_1, t_{-1} for each stage and sends them to the verifier), if it that does not compromise security.
- There is a large amount of identical calculations that both prover and verifier perform on the same data. If only one party can be trusted to perform them, performance can be significantly improved.

Prover $\mathcal{P}$		Verifier $\mathcal{V}$
Inputs:		
$\mathbf{A} \in \mathcal{R}_q^{n \times m}, \mathbf{S} \in \mathcal{S}_B^{m \times k}$		$\mathbf{A}, \mathbf{T}, b, b_1, b_2, l, \vec{g}, \vec{h}, u$
$\mathbf{T} = \mathbf{AS} \in \mathcal{R}_q^{n \times k}$		
$b = \lceil \log(B) \rceil + 1$		
$b_1 = \lceil \log(mdB + d \lVert \mathbf{f} \rVert_\infty) \rceil$		
$b_2 = \lceil \log(q) \rceil$		
$l = mkdb + nk(2d-1)b_1$ $+ nk(d-1)b_2$		
$\vec{g}, \vec{h} \in G^l, u \in G$		
$\mathbf{R}_2 = (\mathbf{T} - \mathbf{AS})/\mathbf{f}$ over $\mathbb{Z}_q[X]$		
$\mathbf{R}_1 = (\mathbf{T} - \mathbf{AS} - \mathbf{f}\mathbf{R}_2)/q$ over $\mathbb{Z}[X]$		
$\vec{s} = \text{Serialize}(\mathbf{S}) \in \mathbb{Z}^{mkd}$		
$\vec{r}_1 = \text{Serialize}(\mathbf{R}_1) \in \mathbb{Z}^{nk(2d-1)}$		
$\vec{r}_2 = \text{Serialize}(\mathbf{R}_2) \in \mathbb{Z}^{nk(d-1)}$	P1	
$\vec{s}_1 = \text{Binary}_b(\vec{s}) \parallel \text{Binary}_{b_1}(\vec{r}_1)$ $\parallel \text{Binary}_{b_2}(\vec{r}_2)$		
$\vec{s}_2 = \vec{s}_1 + \vec{1} \in \mathbb{Z}_2^l$ (XOR)		
$\rho \xleftarrow{\$} \mathbb{Z}_p$		
$w = \vec{g}^{\vec{s}_2} \vec{h}^{\vec{s}_1} u^\rho$	$\xrightarrow{w}$	
	V1	$\alpha \xleftarrow{\$} \mathbb{Z}_p^\times, \vec{\beta} \xleftarrow{\$} (\mathbb{Z}_p^\times)^k, \vec{\gamma} \xleftarrow{\$} (\mathbb{Z}_p^\times)^n$
	$\xleftarrow{\alpha, \vec{\beta}, \vec{\gamma}, \vec{\varphi}, \psi}$	$\vec{\varphi} \xleftarrow{\$} (\mathbb{Z}_p^\times)^l, \psi \xleftarrow{\$} \mathbb{Z}_p^\times$
$\vec{g}' = \vec{g}^{\vec{\varphi}^{-1}}$		$\vec{g}' = \vec{g}^{\vec{\varphi}^{-1}}$
$\vec{v} = \mathbf{A}(\alpha)^T \vec{\gamma} \otimes \vec{\beta} \otimes \vec{\alpha}_d \otimes \vec{2}_b$ $\parallel q\vec{\gamma} \otimes \vec{\beta} \otimes \vec{\alpha}_{2d-1} \otimes \vec{2}_{b_1}$ $\parallel \mathbf{f}(\alpha)\vec{\gamma} \otimes \vec{\beta} \otimes \vec{\alpha}_{d-1} \otimes \vec{2}_{b_2} \in \mathbb{Z}_p^l$	P2	$\vec{v} = \mathbf{A}(\alpha)^T \vec{\gamma} \otimes \vec{\beta} \otimes \vec{\alpha}_d \otimes \vec{2}_b$ $\parallel q\vec{\gamma} \otimes \vec{\beta} \otimes \vec{\alpha}_{2d-1} \otimes \vec{2}_{b_1}$ $\parallel \mathbf{f}(\alpha)\vec{\gamma} \otimes \vec{\beta} \otimes \vec{\alpha}_{d-1} \otimes \vec{2}_{b_2}$
$t = w(\vec{g}')^{\vec{v} + \psi\vec{\varphi}} \vec{h}^\psi$	V2	$t = w(\vec{g}')^{\vec{v} + \psi\vec{\varphi}} \vec{h}^\psi$
$\vec{v}_1 = \vec{v} + \vec{\varphi} \circ \vec{s}_2 + \psi\vec{\varphi}$		
$\vec{v}_2 = \vec{s}_1 + \psi\vec{1}$		
$x = \langle \vec{v}_1, \vec{v}_2 \rangle$		$x = \vec{\gamma}^T \mathbf{T}(\alpha)\vec{\beta} + \psi \left\langle \vec{v}, \vec{1} \right\rangle$ $+ (\psi + \psi^2)\langle \vec{\varphi}, \vec{1} \rangle \in \mathbb{Z}_p$

The parties run the zero-knowledge inner product proof $\rhd\Pi_{\langle \cdot,\cdot \rangle}(\vec{g}', \vec{h}, u, t, x; \vec{v}_1, \vec{v}_2, \rho)$ and the verifier $\mathcal{V}$ accepts if he accepts in $\rhd\Pi_{\langle \cdot,\cdot \rangle}(\cdot;\cdot)$.

Inner product call

Figure 1: Main protocol



Prover $\mathcal{P}$ Verifier $\mathcal{V}$

Inputs:

$\vec{g}, \vec{h} \in G^l; u \in G$ $\vec{g}, \vec{h}, u, t, x$

$\vec{v}_1, \vec{v}_2 \in \mathbb{Z}_p^l; \rho \in \mathbb{Z}_p$

$t = \vec{g}^{\vec{v}_1} \vec{h}^{\vec{v}_2} u^\rho$

$x = \langle \vec{v}_1, \vec{v}_2 \rangle$

$\xleftarrow{\quad a \quad}$ $a \xleftarrow{\$} G$

$t' = ta^x$ $t' = ta^x$

The parties run $(g, h, t''; v_1, v_2, \rho') = \text{FOLDING}(\vec{g}, \vec{h}, a, u, t'; \vec{v}_1, \vec{v}_2, \rho)$ where the secrets $v_1, v_2, \rho' \in \mathbb{Z}_p$ are such that $t'' = g^{v_1} h^{v_2} a^{v_1 v_2} u^{\rho'}$.

$y_1, y_2, \sigma, \sigma' \xleftarrow{\$} \mathbb{Z}_p$

$w = g^{y_1} h^{y_2} a^{y_1 v_2 + y_2 v_1} u^\sigma$

$w' = a^{y_1 y_2} u^{\sigma'}$ $\xrightarrow{\quad w, w' \quad}$

$\xleftarrow{\quad c \quad}$ $c \xleftarrow{\$} \mathbb{Z}_p^\times$

$z_1 = y_1 + cv_1$

$z_2 = y_2 + cv_2$

$\tau = c\rho' + \sigma + c^{-1}\sigma'$ $\xrightarrow{\quad z_1, z_2, \tau \quad}$

$(t'')^c w (w')^{c^{-1}} \stackrel{?}{=} g^{z_1} h^{z_2} a^{c^{-1} z_1 z_2} u^\tau$

Figure 2: Inner product protocol

Prover $\mathcal{P}$	Verifier $\mathcal{V}$
Inputs:	
$\vec{g}, \vec{h} \in G^l; a, u \in G$	$\vec{g}, \vec{h}, a, u, t$
$\vec{v}_1, \vec{v}_2 \in \mathbb{Z}_p^l; \rho \in \mathbb{Z}_p$	
$t = \vec{g}^{\vec{v}_1} \vec{h}^{\vec{v}_2} a^{\langle \vec{v}_1, \vec{v}_2 \rangle} u^{\rho}$	
Outputs:	
$g, h \in G$	g, h, t'
$v_1, v_2, \rho' \in \mathbb{Z}_p$	
$t' = g^{v_1} h^{v_2} a^{v_1 v_2} u^{\rho'}$	

If $l > 1$, define $l' = \frac{l}{2}$ and write $\vec{g} = \left(\begin{smallmatrix} \vec{g}_t \\ \vec{g}_b \end{smallmatrix}\right)$, $\vec{h} = \left(\begin{smallmatrix} \vec{h}_t \\ \vec{h}_b \end{smallmatrix}\right)$, $\vec{v}_i = \left(\begin{smallmatrix} \vec{v}_{i,t} \\ \vec{v}_{i,b} \end{smallmatrix}\right)$, where $\vec{g}_j, \vec{h}_j, \vec{v}_{i,j} \in G^{l'}$ for $i = 1, 2$, $j = t, b$. Then,

$$\left.\begin{aligned} &\sigma_{-1}, \sigma_1 \xleftarrow{\$} \mathbb{Z}_p \\ &t_{-1} = \vec{g}_t^{\vec{v}_{1,b}} \vec{h}_b^{\vec{v}_{2,t}} a^{\langle \vec{v}_{1,b}, \vec{v}_{2,t} \rangle} u^{\sigma_{-1}} \\ &t_1 = \vec{g}_b^{\vec{v}_{1,t}} \vec{h}_t^{\vec{v}_{2,b}} a^{\langle \vec{v}_{1,t}, \vec{v}_{2,b} \rangle} u^{\sigma_1} \end{aligned}\right) \text{P1}$$

$$\xrightarrow{t_{-1}, t_1}$$

$$\xleftarrow{c} \qquad \text{V1}\left(c \xleftarrow{\$} \mathbb{Z}_p^{\times} \right.$$

$$\left.\begin{aligned} &\vec{v}_1' = \vec{v}_{1,t} + c^{-1} \vec{v}_{1,b} \\ &\vec{v}_2' = \vec{v}_{2,t} + c \vec{v}_{2,b} \\ &\rho'' = c^{-1} \sigma_{-1} + \rho + c \sigma_1 \end{aligned}\right) \text{P2}$$

and both parties compute $\vec{g}' = \vec{g}_t \circ \vec{g}_b^c$, $\vec{h}' = \vec{h}_t \circ \vec{h}_b^{c^{-1}}$ and $t'' = t_{-1}^{c^{-1}} t t_1^c$. They recursively run $(g, h, t'; v_1, v_2, \rho') = \text{FOLDING}(\vec{g}', \vec{h}', a, u, t''; \vec{v}_1', \vec{v}_2', \rho'')$ where $\mathcal{P}$ knows $\vec{v}_1'$, $\vec{v}_2'$, ρ'' such that $t'' = (\vec{g}')^{\vec{v}_1'} (\vec{h}')^{\vec{v}_2'} a^{\langle \vec{v}_1', \vec{v}_2' \rangle} u^{\rho''}$.

fold_commitment()
(part of P2 and V2)

Else $g = \vec{g}, h = \vec{h} \in G$, and $\mathcal{P}$ knows $v_1 = \vec{v}_1, v_2 = \vec{v}_2, \rho' = \rho \in \mathbb{Z}_p$, such that $t' = t = g^{v_1} h^{v_2} a^{v_1 v_2} u^{\rho'}$.

Figure 3: Folding protocol