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	size (compressed points)
Shared randomness (to verifier)	19.9 Mb	6.13 Mb
Random challenge (to prover)	3.05 Mb	3.05 Mb
Folding info (to verifier)	208 b × 17	64 b × 17
Folding info (to prover)	32 b × 17	32 b × 17
Inner product info (to verifier)	208 b	64 b
Inner product info (to prover)	32 b	32 b
Inner product info (to verifier)	96 b	96 b

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

main	inner product	folding	time	main	inner product	folding	time
P1			702ms				
				V1			2.09ms
P2			25.9s	V2			25.6s
					V1		86.0µs
	P1		7.17ms		V2		7.99ms
		P1	21.5s				
						V1	31.5µs
		P2	16.6s			V2	16.6s
	P2		519µs				
					V3		2.29µs
	P3		118µs				
					V4 (check happens here)		759µs
Prover time			$\sim$ 64 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			807ms				
				V1			2.09ms
P2			5.36s	V2			5.38s
					V1		109µs
	P1		32.7ms		V2		20.1ms
		P1	5.15s				
						V1	31.8µs
		P2	4.36s			V2	4.30s
	P2		611µs				
					V3		2.95µs
	P3		133µs				
					V4 (check happens here)		906µs
Prover time			$\sim$ 14.9 s	Verifier time			$\sim$ 9.7 s

Totals

	i7 @ 2.6 GHz, 1 thread	i7 @ 2.6 GHz, 6 threads
Prover time	70s	14.9s
Verifier time	47s	9.7s
Encryption time	1.18ms	=
Decryption time	607µs	=
Initial proof generation	16.0s	3.23s
Prover transfers	20 Mb (non-compressed) / 6 Mb (compressed)	
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			535ms				
				V1			5.25ms
P2			9.14s	V2			8.90s
					V1		20.0µs
	P1		50.9ms		V2		76.1ms
		P1	9.70s				
						V1	29.0µs
		P2	15.2s			V2	14.8s
	P2		234µs				
					V3		2.42µs
	P3		31.2µs				
					V4 (check happens here)		309µs
Prover time			~ 34.6 s	Verifier time			~ 23.7 s

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

main	inner product	folding	time	main	inner product	folding	time
P1			564ms				
				V1			5.34ms
P2			2.09s	V2			2.10s
					V1		20.6µs
	P1		25.7ms		V2		33.4ms
		P1	2.37s				
						V1	30.0µs
		P2	3.15s			V2	3.08s
	P2		273µs				
					V3		3.70µs
	P3		37.2µs				
					V4 (check happens here)		329µs
Prover time			~ 8.2 s	Verifier time			~ 5.2 s

Totals

	i7 @ 2.6 GHz, 1 thread	i7 @ 2.6 GHz, 6 threads
Prover time	34.6s	8.2s
Verifier time	23.7s	5.2s
Encryption time	1.36ms	=
Initial proof generation	2.15s	434ms
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 $\mathcal{P}$

Inputs:

$\mathbf{A} \in \mathcal{R}_q^{n \times m}, \mathbf{S} \in \mathcal{S}_B^{m \times k}$

$\mathbf{T} = \mathbf{AS} \in \mathcal{R}_q^{n \times k}$

$b = \lceil \log(B) \rceil + 1$

$b_1 = \lceil \log(mdB + d \|\mathbf{f}\|_\infty) \rceil$

$b_2 = \lceil \log(q) \rceil$

$l = mkdb + nk(2d-1)b_1$
$\quad + nk(d-1)b_2$

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

Verifier $\mathcal{V}$

$\mathbf{A}, \mathbf{T}, b, b_1, b_2, l, \vec{g}, \vec{h}, u$

$\mathbf{R}_2 = (\mathbf{T} - \mathbf{AS})/\mathbf{f}$ over $\mathbb{Z}_q[X]$

$\mathbf{R}_1 = (\mathbf{T} - \mathbf{AS} - \mathbf{fR}_2)/q$ over $\mathbb{Z}[X]$

$\vec{s} = \text{Serialize}(\mathbf{S}) \in \mathbb{Z}^{mkd}$

$\vec{r}_1 = \text{Serialize}(\mathbf{R}_1) \in \mathbb{Z}^{nk(2d-1)}$

$\vec{r}_2 = \text{Serialize}(\mathbf{R}_2) \in \mathbb{Z}^{nk(d-1)}$

$\vec{s}_1 = \text{Binary}_b(\vec{s}) \parallel \text{Binary}_{b_1}(\vec{r}_1)$
$\quad \parallel \text{Binary}_{b_2}(\vec{r}_2)$

$\vec{s}_2 = \vec{s}_1 + \vec{1} \in \mathbb{Z}_2^l$ (XOR)

$\rho \xleftarrow{\$} \mathbb{Z}_p$

$w = \vec{g}^{\vec{s}_2} \vec{h}^{\vec{s}_1} u^\rho$

P1

$\xrightarrow{w}$

V1

$\alpha \xleftarrow{\$} \mathbb{Z}_p^\times, \vec{\beta} \xleftarrow{\$} (\mathbb{Z}_p^\times)^k, \vec{\gamma} \xleftarrow{\$} (\mathbb{Z}_p^\times)^n$

$\vec{\varphi} \xleftarrow{\$} (\mathbb{Z}_p^\times)^l, \psi \xleftarrow{\$} \mathbb{Z}_p^\times$

$\xleftarrow{\alpha, \vec{\beta}, \vec{\gamma}, \vec{\varphi}, \psi}$

$\vec{g}' = \vec{g}^{\vec{\varphi}^{-1}}$

$\vec{v} = \mathbf{A}(\alpha)^T \vec{\gamma} \otimes \vec{\beta} \otimes \vec{\alpha}_d \otimes \vec{2}_b$
$\quad \parallel q\vec{\gamma} \otimes \vec{\beta} \otimes \vec{\alpha}_{2d-1} \otimes \vec{2}_{b_1}$
$\quad \parallel \mathbf{f}(\alpha)\vec{\gamma} \otimes \vec{\beta} \otimes \vec{\alpha}_{d-1} \otimes \vec{2}_{b_2} \in \mathbb{Z}_p^l$

$t = w(\vec{g}')^{\vec{v} + \psi\vec{\varphi}} \vec{h}^\psi$

$\vec{v}_1 = \vec{v} + \vec{\varphi} \circ \vec{s}_2 + \psi\vec{\varphi}$

$\vec{v}_2 = \vec{s}_1 + \psi\vec{1}$

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

P2

V2

$\vec{g}' = \vec{g}^{\vec{\varphi}^{-1}}$

$\vec{v} = \mathbf{A}(\alpha)^T \vec{\gamma} \otimes \vec{\beta} \otimes \vec{\alpha}_d \otimes \vec{2}_b$
$\quad \parallel q\vec{\gamma} \otimes \vec{\beta} \otimes \vec{\alpha}_{2d-1} \otimes \vec{2}_{b_1}$
$\quad \parallel \mathbf{f}(\alpha)\vec{\gamma} \otimes \vec{\beta} \otimes \vec{\alpha}_{d-1} \otimes \vec{2}_{b_2}$

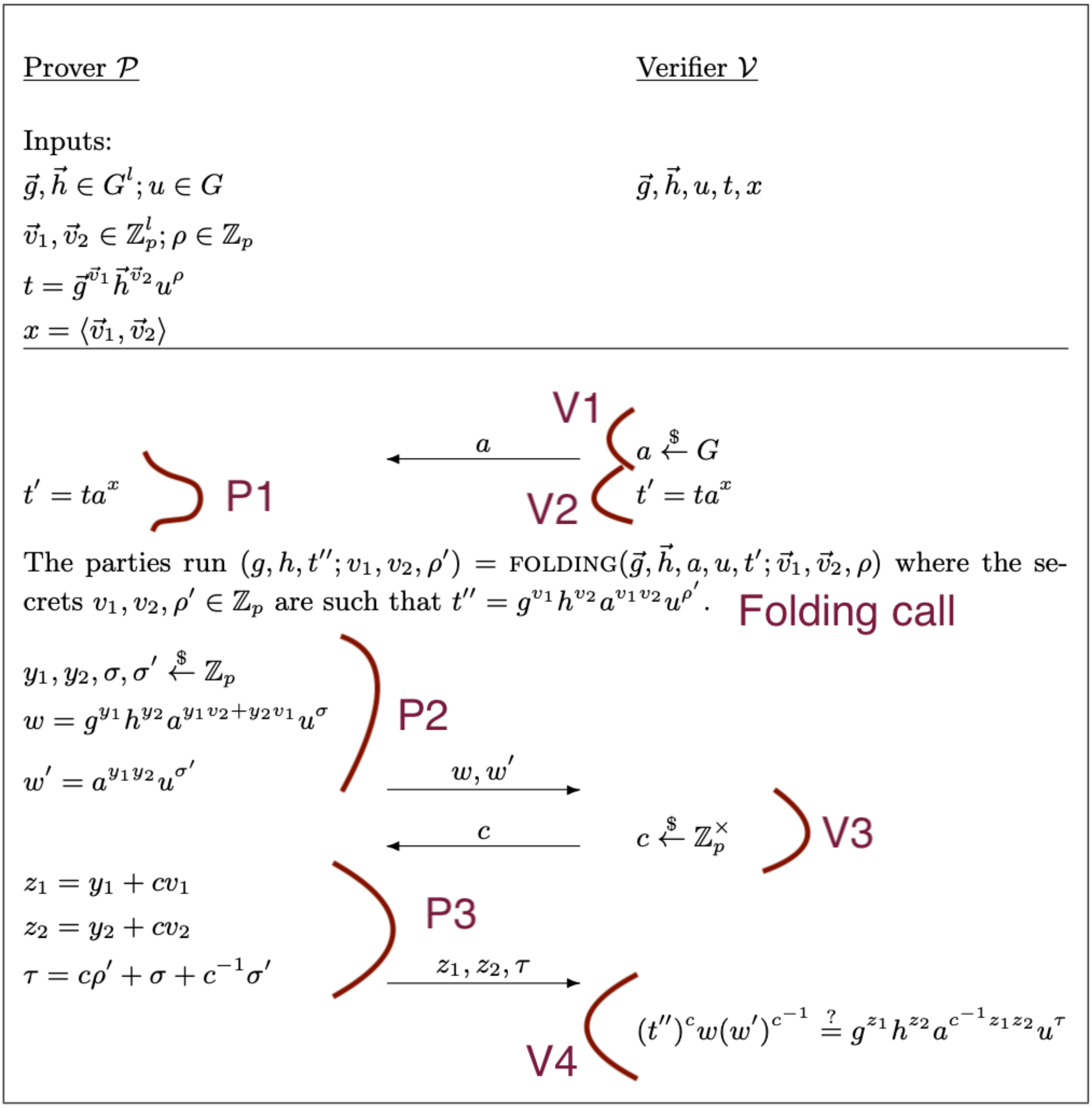
$t = w(\vec{g}')^{\vec{v} + \psi\vec{\varphi}} \vec{h}^\psi$

$x = \vec{\gamma}^T \mathbf{T}(\alpha)\vec{\beta} + \psi \left\langle \vec{v}, \vec{1} \right\rangle$
$\quad + (\psi + \psi^2)\langle \vec{\varphi}, \vec{1} \rangle \in \mathbb{Z}_p$

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

Inner product call

Figure 1: Main protocol



<u>Prover $\mathcal{P}$</u>

<u>Verifier $\mathcal{V}$</u>

Inputs:

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

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

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

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

Verifier inputs: $\vec{g}, \vec{h}, u, t, x$

Verifier: $a \xleftarrow{\$} G$

$\xleftarrow{\quad a \quad}$

Verifier: $t' = ta^x$

Prover: $t' = ta^x$

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

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

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

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

$\xrightarrow{\quad w, w' \quad}$

$\xleftarrow{\quad c \quad}$

Verifier: $c \xleftarrow{\$} \mathbb{Z}_p^\times$

$z_1 = y_1 + cv_1$

$z_2 = y_2 + cv_2$

$\tau = c\rho' + \sigma + c^{-1}\sigma'$

$\xrightarrow{\quad z_1, z_2, \tau \quad}$

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

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 $\vec{g} = \binom{\vec{g}_t}{\vec{g}_b}$, $\vec{h} = \binom{\vec{h}_t}{\vec{h}_b}$, $\vec{v}_i = \binom{\vec{v}_{i,t}}{\vec{v}_{i,b}}$, where $\vec{g}_j, \vec{h}_j, \vec{v}_{i,j} \in G^{l'}$ for $i = 1, 2$, $j = t, b$. Then,

$$\sigma_{-1}, \sigma_1 \overset{\$}{\leftarrow} \mathbb{Z}_p$$
$$t_{-1} = \vec{g}_t^{\vec{v}_{1,b}} \vec{h}_b^{\vec{v}_{2,t}} a^{\langle \vec{v}_{1,b}, \vec{v}_{2,t} \rangle} u^{\sigma_{-1}}$$
$$t_1 = \vec{g}_b^{\vec{v}_{1,t}} \vec{h}_t^{\vec{v}_{2,b}} a^{\langle \vec{v}_{1,t}, \vec{v}_{2,b} \rangle} u^{\sigma_1}$$

P1

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

$$\xleftarrow{c}$$

V1 $c \overset{\$}{\leftarrow} \mathbb{Z}_p^{\times}$

$$\vec{v}_1' = \vec{v}_{1,t} + c^{-1} \vec{v}_{1,b}$$
$$\vec{v}_2' = \vec{v}_{2,t} + c \vec{v}_{2,b}$$
$$\rho'' = c^{-1} \sigma_{-1} + \rho + c \sigma_1$$

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