Bern University of Applied Sciences

Performance of Shuffling: Taking it to the Limits

Rolf Haenni, Philipp Locher

Voting'20 @ FC'20, Kota Kinabalu, February 14, 2020

Outline

- Introduction
- Optimization Techniques
- Application to Wikström's Shuffle Proof
- Conclusion

Motivation

- Voting protocols often depend on verifiable re-encryption mix-nets
- For large electorates, mixing the submitted encrypted votes may become a performance bottleneck
- Example:
  - $N = 100\,000$ ciphertexts, $m = 4$ mix nodes
  - $10N = 1\,000\,000$ exponentiations per mix node (Wikström)
  - $40N = 40\,000\,000$ exponentiations in total
  - Similar for proof verification
- Assuming that modular exponentiation takes 9 milliseconds for 3072-bits numbers, we get 10 hours of computation

Outline

- Introduction
- Optimization Techniques
- Application to Wikström's Shuffle Proof
- Conclusion

General Exponentiation

- Group $(\mathcal{G}, \cdot, ^{-1}, 1)$ of order q
- On inputs $b \in \mathcal{G}$ and $e \in \mathbb{Z}_q$, compute
$$z = \mathrm{Exp}(b, e) = b^e$$
- Exponent size: $\ell = \log_2 e$
- Sliding-Window Algorithm: $1 \leq k \leq \ell$ (window size)

Multiplications	112	128	224	256	2048	3072
$M_k(\ell) = 2^{k-1} + \ell + \frac{\ell}{k+2}$	$k = 3$		$k = 4$		$k = 6$	$k = 7$

- Example: $M_7(3072) = 3477$

Product Exponentiation

- On inputs $\mathbf{b} = (b_1, \ldots, b_N) \in \mathcal{G}^N$ and $\mathbf{e} = (e_1, \ldots, e_N) \in \mathbb{Z}^N$, compute

$$z = \mathsf{ProductExp}(\mathbf{b}, \mathbf{e}) = \prod_{i=1}^{n} b_i^{e_i}$$

- Maximal exponent size: $\ell = \max_{i=1}^{N} \log_2 e_i$
- Algorithm 2 from last year's paper: $1 \leq m \leq N$ (subtask size)

Multiplications	112	128	224	256	2048	3072
$\widetilde{M}_m(\ell, N) = \dfrac{2^m + \ell}{m} + \dfrac{\ell}{N}$	$m = 5$		$m = 6$		$m = 9$	

- Example: $\widetilde{M}_9(3072, \text{large } N) = 396$
- Relative speedup: 8.8

Fixed-Base Exponentiation

- On inputs $b \in \mathcal{G}$ and $\mathbf{e} = (e_1, \ldots, e_N) \in \mathbb{Z}^N$, compute
$$\mathbf{z} = \mathrm{FixedBaseExp}(b, \mathbf{e}) = (b^{e_1}, \ldots, b^{e_N})$$
- Maximal exponent size: $\ell = \max_{i=1}^{N} \log_2 e_i$
- Algorithm 3.2 from last year's paper: $1 \le k \le \ell$, $1 \le m \le \ell/k$
$$\widetilde{M}_{k,m}(\ell, N) = \frac{\ell}{N}\left(\frac{2^m}{km} + 1\right) + \frac{\ell}{m} + k$$
- Examples:
$$\widetilde{M}_{32,12}(3072, 1000) = 320 \text{ (relative speedup: 10.7)}$$
$$\widetilde{M}_{19,18}(3072, 100\,000) = 210 \text{ (relative speedup: 16.6)}$$
$$\widetilde{M}_{11,20}(3072, 1\,000\,000) = 176 \text{ (relative speedup: 19.8)}$$

Batch Verification: General Case

- On inputs $\mathbf{z} = (z_1, \ldots, z_N)$, $\mathbf{b} = (b_1, \ldots, b_N)$, $\mathbf{e} = (e_1, \ldots, e_N)$, compute

$$\mathrm{BatchVerif}(\mathbf{z}, \mathbf{b}, \mathbf{e}) = \bigwedge_{i=1}^{N} \left[z_i = b_i^{e_i} \right] \in \{0, 1\}$$

- Small Exponent Test (SET):
  - Pick s-bits values $s_i \in_R \{0, \ldots, 2^s - 1\}$ at random
  - Compute ℓ-bits values $s'_i = s_i e_i \bmod q$
  - Let $\mathbf{s} = (s_1, \ldots, s_N)$, $\mathbf{s}' = (s'_1, \ldots, s'_N)$

- Perform the following check:

$$\mathrm{ProductExp}(\mathbf{z}, \mathbf{s}) \stackrel{?}{=} \mathrm{ProductExp}(\mathbf{b}, \mathbf{s}')$$

- Failure probability: $P(\exists\, z_i \neq b_i^{e_i}) = 2^{-s}$

- Pre-conditions: prime-order q, group membership $z_i \in \mathcal{G}$

Batch Verification: Special Cases

- For fixed base $\mathbf{b} = (b, \ldots, b)$, $s' = \sum_{i=1}^{N} s'_i \bmod q$, check

$$\mathsf{ProductExp}(\mathbf{z}, \mathbf{s}) \stackrel{?}{=} \mathsf{Exp}(b, s')$$

- For fixed exponent $\mathbf{e} = (e, \ldots, e)$, check

$$\mathsf{ProductExp}(\mathbf{z}, \mathbf{s}) \stackrel{?}{=} \mathsf{Exp}(\mathsf{ProductExp}(\mathbf{b}, \mathbf{s}), e)$$

- Example: $\ell = 3072$ and $s = 128$
  - 30 multiplications for $\mathsf{ProductExp}(\cdot, \mathbf{s})$
  - 396 multiplications for $\mathsf{ProductExp}(\cdot, \mathbf{s}')$
  - $3477/N$ mulitplications for $\mathsf{Exp}(\cdot, s')$ and $\mathsf{Exp}(\cdot, e)$

General Case	Fixed Base	Fixed Exponent
426	$30 + 3477/N$	$60 + 3477/N$

Group Membership Tests (GMT)

- General group: $z \in \mathcal{G}$ iff $z^q = 1$
- Integers modulo prime p: $z \in \mathbb{Z}_p^*$ iff
$$z \in \{1, \ldots, p-1\}$$
- Elliptic curve: $z = (x, y) \in E(\mathbb{F}_p)$ iff
$$x, y \in \{0, \ldots, p-1\} \text{ and } y^2 = x^3 + ax + b$$
- Quadratic residues modulo $p = 2q + 1$: $z \in \mathbb{G}_q$ iff
$$\left(\frac{z}{p}\right) = 1$$

  Remark: for $\ell = 3072$, computing the Jacobi symbol is approx. 20 times faster than exponentiation

GMT using Square Root Witnesses

- For $p = 2q + 1$, every $z \in \mathbb{G}_q$ has exactly two square roots $\sqrt{z} = \pm x^{\frac{q+1}{2}} \bmod p$, whereas $x \notin \mathbb{G}_q$ has no square roots
- By presenting $\sqrt{x}$ as a group membership witness for x, $x \in \mathbb{G}_q$ can be tested using a single multiplication
- Therefore, representing elements $x \in \mathbb{G}_q$ by pairs $\hat{x} = (\sqrt{x}, x)$ enables an efficient membership test for $\mathbb{G}_q$
- Note that group operations can be conducted on the square roots modulo p:

$$\sqrt{xy} = \sqrt{x}\sqrt{y}, \quad \sqrt{x^e} = \sqrt{x}^{\,e}, \quad \sqrt{x^{-1}} = \sqrt{x}^{\,-1}$$

- By computing x in $\hat{x} = (\sqrt{x}, x)$ lazily (only when needed), GMT in $\mathbb{G}_q \subset \mathbb{Z}_p^*$ comes at almost no cost

Outline

- Introduction
- Optimization Techniques
- Application to Wikström's Shuffle Proof
- Conclusion

Re-Encryption Shuffle

- Parameters: security strength λ, group size ℓ (bits)
- Two inputs:

$$\begin{aligned} \mathbf{e} &= \text{input list of (ElGamal) ciphertexts} \\ pk &= \text{encryption public key} \end{aligned}$$

- Two outputs:

$$\begin{aligned} \tilde{\mathbf{e}} &= \text{permuted list of re-encrypted ciphertexts} \\ \pi &= \text{non-interative zero-knowledge proof} \end{aligned}$$

- Three main algorithms:

$$\begin{aligned} (\tilde{\mathbf{e}}, \tilde{\mathbf{r}}, \psi) &\leftarrow \mathsf{GenShuffle}(\mathbf{e}, pk) \\ \pi &\leftarrow \mathsf{GenProof}(\mathbf{e}, \tilde{\mathbf{e}}, \tilde{\mathbf{r}}, \psi, pk) \\ \mathit{true}/\mathit{false} &\leftarrow \mathsf{CheckProof}(\pi, \mathbf{e}, \tilde{\mathbf{e}}, pk) \end{aligned}$$

Wikström's Shuffle Algorithms

Algorithm: $\mathsf{GenShuffle}(\boldsymbol{e}, pk)$

Input: ElGamal ciphertexts $\boldsymbol{e} = (e_1, \ldots, e_N)$, $e_i = (a_i, b_i) \in \mathcal{G}^2$
Encryption key $pk \in \mathcal{G}$

$\psi \leftarrow \mathsf{GenPermutation}(N)$
for $i = 1, \ldots, N$ **do**
  $\tilde{r}_i \in_R \mathbb{Z}_q$
  $\tilde{a}_i \leftarrow a_i \cdot pk^{\tilde{r}_i}$
  $\tilde{b}_i \leftarrow b_i \cdot g^{\tilde{r}_i}$
  $\tilde{e}_i \leftarrow (\tilde{a}_i, \tilde{b}_i)$
$\tilde{\boldsymbol{e}} \leftarrow (\tilde{e}_{j_1}, \ldots, \tilde{e}_{j_N})$
$\tilde{\boldsymbol{r}} \leftarrow (\tilde{r}_1, \ldots, \tilde{r}_N)$
return $(\tilde{\boldsymbol{e}}, \tilde{\boldsymbol{r}}, \psi)$ // $\tilde{\boldsymbol{e}} \in (\mathcal{G}^2)^N$, $\tilde{\boldsymbol{r}} \in \mathbb{Z}_q^N$, $\psi \in \Psi_N$

Algorithm: $\mathsf{CheckProof}(\pi, \boldsymbol{e}, \tilde{\boldsymbol{e}}, pk)$

Input: Shuffle proof $\pi = (t, s, \boldsymbol{c}, \hat{\boldsymbol{c}})$
– $t = (t_1, t_2, t_3, (t_{4,1}, t_{4,2}), (\hat{t}_1, \ldots, \hat{t}_N)) \in \mathcal{G} \times \mathcal{G} \times \mathcal{G} \times \mathcal{G}^2 \times \mathcal{G}^N$
– $s = (s_1, s_2, s_3, s_4, (\hat{s}_1, \ldots, \hat{s}_N), (\tilde{s}_1, \ldots, \tilde{s}_N)) \in \mathbb{Z}_q \times \mathbb{Z}_q \times \mathbb{Z}_q \times \mathbb{Z}_q \times \mathbb{Z}_q^N \times \mathbb{Z}_q^N$
– $\boldsymbol{c} = (c_1, \ldots, c_N) \in \mathcal{G}^N$, $\hat{\boldsymbol{c}} = (\hat{c}_1, \ldots, \hat{c}_N) \in \mathcal{G}^N$
ElGamal ciphertexts $\boldsymbol{e} = (e_1, \ldots, e_N)$, $e_i = (a_i, b_i) \in \mathcal{G}^2$
Shuffled ElGamal ciphertexts $\tilde{\boldsymbol{e}} = (\tilde{e}_1, \ldots, \tilde{e}_N)$, $\tilde{e}_i = (\tilde{a}_i, \tilde{b}_i) \in \mathcal{G}^2$
Encryption key $pk \in \mathcal{G}$

for $i = 1, \ldots, N$ **do**
  $u_i \leftarrow \mathsf{Hash}((\boldsymbol{e}, \tilde{\boldsymbol{e}}, \boldsymbol{c}), i)$
$\hat{c}_0 \leftarrow h$
$\bar{c} \leftarrow \prod_{i=1}^N c_i / \prod_{i=1}^N h_i$
$u \leftarrow \prod_{i=1}^N u_i \bmod q$
$\hat{c} \leftarrow \hat{c}_N \cdot h^{-u}$
$\tilde{c} \leftarrow \prod_{i=1}^N c_i^{u_i}$
$\tilde{a} \leftarrow \prod_{i=1}^N a_i^{u_i}$
$\tilde{b} \leftarrow \prod_{i=1}^N b_i^{u_i}$
$c \leftarrow \mathsf{Hash}(\boldsymbol{e}, \tilde{\boldsymbol{e}}, \boldsymbol{c}, \hat{\boldsymbol{c}}, pk, t)$
for $i = 1, \ldots, N$ **do**
  $\hat{t}'_i \leftarrow \hat{c}_i^c \cdot g^{\hat{s}_i} \cdot \hat{c}_{i-1}^{\tilde{s}_i}$
$t'_1 \leftarrow \bar{c}^{\,c} \cdot g^{s_1}$
$t'_2 \leftarrow \hat{c}^{\,c} \cdot g^{s_2}$
$t'_3 \leftarrow \tilde{c}^{\,c} \cdot g^{s_3} \cdot \prod_{i=1}^N h_i^{\tilde{s}_i}$
$t'_{4,1} \leftarrow \tilde{a}^c \cdot pk^{-s_4} \cdot \prod_{i=1}^N \tilde{a}_i^{\tilde{s}_i}$
$t'_{4,2} \leftarrow \tilde{b}^c \cdot g^{-s_4} \cdot \prod_{i=1}^N \tilde{b}_i^{\tilde{s}_i}$
return
$(t_1 = t'_1) \wedge (t_2 = t'_2) \wedge (t_3 = t'_3) \wedge (t_{4,1} = t'_{4,1}) \wedge (t_{4,2} = t'_{4,2}) \wedge \left[\bigwedge_{i=1}^N (\hat{t}_i = \hat{t}'_i)\right]$

Algorithm: $\mathsf{GenProof}(\boldsymbol{e}, \tilde{\boldsymbol{e}}, \tilde{\boldsymbol{r}}, \psi, pk)$

Input: ElGamal ciphertexts $\boldsymbol{e} = (e_1, \ldots, e_N)$, $e_i = (a_i, b_i) \in \mathcal{G}^2$
Shuffled ElGamal ciphertexts $\tilde{\boldsymbol{e}} = (\tilde{e}_1, \ldots, \tilde{e}_N)$, $\tilde{e}_i = (\tilde{a}_i, \tilde{b}_i) \in \mathcal{G}^2$
Re-encryption randomizations $\tilde{\boldsymbol{r}} = (\tilde{r}_1, \ldots, \tilde{r}_N)$, $\tilde{r}_i \in \mathbb{Z}_q$
Permutation $\psi = (j_1, \ldots, j_N) \in \Psi_N$
Encryption key $pk \in \mathcal{G}$

for $i = 1, \ldots, N$ **do**
  $r_{j_i} \in_R \mathbb{Z}_q$
  $c_{j_i} \leftarrow h_i \cdot g^{r_{j_i}}$
$\boldsymbol{c} = (c_1, \ldots, c_N)$
for $i = 1, \ldots, N$ **do**
  $u_i \leftarrow \mathsf{Hash}((\boldsymbol{e}, \tilde{\boldsymbol{e}}, \boldsymbol{c}), i)$
$\hat{c}_0 \leftarrow h$
for $i = 1, \ldots, N$ **do**
  $\hat{r}_i \in_R \mathbb{Z}_q$, $\tilde{u}_i \leftarrow u_{j_i}$
  $\hat{c}_i \leftarrow g^{\hat{r}_i} \cdot \hat{c}_{i-1}^{\tilde{u}_i}$
$\hat{\boldsymbol{c}} = (\hat{c}_1, \ldots, \hat{c}_N)$
for $i = 1, \ldots, N$ **do**
  $\hat{\omega}_i \in_R \mathbb{Z}_q$, $\tilde{\omega}_i \in_R \mathbb{Z}_q$
  $\hat{t}_i \leftarrow g^{\hat{\omega}_i} \cdot \hat{c}_{i-1}^{\tilde{\omega}_i}$
$\omega_1 \in_R \mathbb{Z}_q$, $\omega_2 \in_R \mathbb{Z}_q$, $\omega_3 \in_R \mathbb{Z}_q$, $\omega_4 \in_R \mathbb{Z}_q$
$t_1 \leftarrow g^{\omega_1}$
$t_2 \leftarrow g^{\omega_2}$
$t_3 \leftarrow g^{\omega_3} \cdot \prod_{i=1}^N h_i^{\tilde{\omega}_i}$
$t_{4,1} \leftarrow pk^{-\omega_4} \cdot \prod_{i=1}^N \tilde{a}_i^{\tilde{\omega}_i}$
$t_{4,2} \leftarrow g^{-\omega_4} \cdot \prod_{i=1}^N \tilde{b}_i^{\tilde{\omega}_i}$
$t \leftarrow (t_1, t_2, t_3, (t_{4,1}, t_{4,2}), (\hat{t}_1, \ldots, \hat{t}_N))$
$c \leftarrow \mathsf{Hash}(\boldsymbol{e}, \tilde{\boldsymbol{e}}, \boldsymbol{c}, \hat{\boldsymbol{c}}, pk, t)$
$v_N \leftarrow 1$
for $i = N, \ldots, 1$ **do**
  $v_{i-1} \leftarrow \tilde{u}_i v_i \bmod q$
$r \leftarrow \sum_{i=1}^N r_i \bmod q$, $s_1 \leftarrow \omega_1 - c \cdot r \bmod q$
$\hat{r} \leftarrow \sum_{i=1}^N \hat{r}_i v_i \bmod q$, $s_2 \leftarrow \omega_2 - c \cdot \hat{r} \bmod q$
$\bar{r} \leftarrow \sum_{i=1}^N r_i u_i \bmod q$, $s_3 \leftarrow \omega_3 - c \cdot \bar{r} \bmod q$
$\tilde{r} \leftarrow \sum_{i=1}^N \tilde{r}_i u_i \bmod q$, $s_4 \leftarrow \omega_4 - c \cdot \tilde{r} \bmod q$
for $i = 1, \ldots, N$ **do**
  $\hat{s}_i \leftarrow \hat{\omega}_i - c \cdot \hat{r}_i \bmod q$, $\tilde{s}_i \leftarrow \tilde{\omega}_i - c \cdot \tilde{u}_i \bmod q$
$s \leftarrow (s_1, s_2, s_3, s_4, (\hat{s}_1, \ldots, \hat{s}_N), (\tilde{s}_1, \ldots, \tilde{s}_N))$
$\pi \leftarrow (t, s, \boldsymbol{c}, \hat{\boldsymbol{c}})$
return $\pi \in (\mathcal{G} \times \mathcal{G} \times \mathcal{G} \times \mathcal{G}^2 \times \mathcal{G}^N) \times (\mathbb{Z}_q \times \mathbb{Z}_q \times \mathbb{Z}_q \times \mathbb{Z}_q \times \mathbb{Z}_q^N \times \mathbb{Z}_q^N) \times \mathcal{G}^N \times \mathcal{G}^N$

Overview of Modular Exponentiations

Algorithm	Line	Computation	PLE		PRE	FBE		GMT
			ℓ	λ	ℓ	ℓ	b	
GenShuffle	1a	$(a_i, b_i) \in \mathcal{G}^2$	–	–	–	–	–	$2N$
	1b	$pk \in \mathcal{G}$	–	–	–	–	–	1
	5	$\tilde{a}_i \leftarrow a_i \cdot pk^{\tilde{r}_i}$	–	–	–	N	pk	–
	6	$\tilde{b}_i \leftarrow b_i \cdot g^{\tilde{r}_i}$	–	–	–	N	g	–
GenProof	4	$c_{j_i} \leftarrow h_i \cdot g^{r_{j_i}}$	–	–	–	N	g	–
	11	$\hat{c}_i \leftarrow g^{\hat{r}_i} \cdot \hat{c}_{i-1}^{\tilde{u}_i}$	–	N	–	N	g	–
	15	$\hat{t}_i \leftarrow g^{\hat{\omega}_i} \cdot \hat{c}_{i-1}^{\tilde{\omega}_i}$	N	–	–	N	g	–
	17	$t_1 \leftarrow g^{\omega_1}$	–	–	–	1	g	–
	18	$t_2 \leftarrow g^{\omega_2}$	–	–	–	1	g	–
	19	$t_3 \leftarrow g^{\omega_3} \cdot \prod_{i=1}^{N} h_i^{\tilde{\omega}_i}$	–	–	N	1	g	–
	20	$t_{4,1} \leftarrow pk^{-\omega_4} \cdot \prod_{i=1}^{N} \tilde{a}_i^{\tilde{\omega}_i}$	–	–	N	1	pk	–
	21	$t_{4,2} \leftarrow g^{-\omega_4} \cdot \prod_{i=1}^{N} \tilde{b}_i^{\tilde{\omega}_i}$	–	–	N	1	g	–
Total			$10N + 5$					$2N + 1$

PLE = plain, PRE = product, FBE = fixed-base, GMT = group membership

Overview of Modular Exponentiations

Algorithm	Line	Computation	PLE		PRE		FBE		GMT
			ℓ	λ	ℓ	λ	ℓ	b	
CheckProof	1a	$t \in \mathcal{G} \times \mathcal{G} \times \mathcal{G} \times \mathcal{G}^2 \times \mathcal{G}^N$	–	–	–	–	–	–	$N+5$
	1b	$\boldsymbol{c} \in \mathcal{G}^N$, $\hat{\boldsymbol{c}} \in \mathcal{G}^N$	–	–	–	–	–	–	$2N$
	1c	$(a_i, b_i) \in \mathcal{G}^2$, $(\tilde{a}_i, \tilde{b}_i) \in \mathcal{G}^2$	–	–	–	–	–	–	$4N$
	1d	$pk \in \mathcal{G}$	–	–	–	–	–	–	1
	7	$\hat{c} \leftarrow \hat{c}_N \cdot h^{-u}$	–	–	–	–	1	h	–
	8	$\tilde{c} \leftarrow \prod_{i=1}^N c_i^{u_i}$	–	–	–	N	–	–	–
	9	$\tilde{a} \leftarrow \prod_{i=1}^N a_i^{u_i}$	–	–	–	N	–	–	–
	10	$\tilde{b} \leftarrow \prod_{i=1}^N b_i^{u_i}$	–	–	–	N	–	–	–
	13	$\hat{t}'_i \leftarrow \hat{c}_i^{\,c} \cdot g^{\hat{s}_i} \cdot \hat{c}_{i-1}^{\tilde{s}_i}$	N	N	–	–	N	g	–
	14	$t'_1 \leftarrow \bar{c}^{\,c} \cdot g^{s_1}$	–	1	–	–	1	g	–
	15	$t'_2 \leftarrow \hat{c}^{\,c} \cdot g^{s_2}$	–	1	–	–	1	g	–
	16	$t'_3 \leftarrow \tilde{c}^{\,c} \cdot g^{s_3} \cdot \prod_{i=1}^N h_i^{\tilde{s}_i}$	–	1	N	–	1	g	–
	17	$t'_{4,1} \leftarrow \tilde{a}^c \cdot pk^{-s_4} \cdot \prod_{i=1}^N \tilde{a}_i^{\tilde{s}_i}$	–	1	N	–	1	pk	–
	18	$t'_{4,2} \leftarrow \tilde{b}^c \cdot g^{-s_4} \cdot \prod_{i=1}^N \tilde{b}_i^{\tilde{s}_i}$	–	1	N	–	1	g	–
Total			$9N+11$						$7N+6$

Performance Improvement

Not optimized				Partly optimized				
Generate		Verify		Generate		Verify		N
31622	1.00	18221	1.00	6742	0.21	5406	0.30	100
31465	1.00	18027	1.00	6264	0.20	5233	0.29	1 000
31450	1.00	18007	1.00	5971	0.19	5162	0.29	10 000
31448	1.00	18007	1.00	5782	0.18	5117	0.28	100 000
31448	1.00	18005	1.00	5640	0.18	5083	0.28	1 000 000

$\lambda = 128$, $\ell = 3072$ bits

Performance Improvement

Not optimized				Partly optimized				
Generate		Verify		Generate		Verify		N
31622	1.00	18221	1.00	6742	0.21	5406	0.30	100
31465	1.00	18027	1.00	6264	0.20	5233	0.29	1 000
31450	1.00	18007	1.00	5971	0.19	5162	0.29	10 000
31448	1.00	18007	1.00	5782	0.18	5117	0.28	100 000
31448	1.00	18005	1.00	5640	0.18	5083	0.28	1 000 000

$\lambda = 128$, $\ell = 3072$ bits

Performance Improvement

Not optimized				Partly optimized				
Generate		Verify		Generate		Verify		N
2926	1.00	2184	1.00	822	0.28	768	0.35	100
2913	1.00	2161	1.00	763	0.26	742	0.34	1 000
2911	1.00	2158	1.00	725	0.25	731	0.34	10 000
2911	1.00	2158	1.00	699	0.24	725	0.33	100 000
2911	1.00	2158	1.00	683	0.23	721	0.33	1 000 000

$\lambda = 128$, $\ell = 256$ bits

Performance Improvement

Not optimized				Partly optimized				
Generate		Verify		Generate		Verify		N
2926	1.00	2184	1.00	822	0.28	768	0.35	100
2913	1.00	2161	1.00	763	0.26	742	0.34	1 000
2911	1.00	2158	1.00	725	0.25	731	0.34	10 000
2911	1.00	2158	1.00	699	0.24	725	0.33	100 000
2911	1.00	2158	1.00	683	0.23	721	0.33	1 000 000

$\lambda = 128$, $\ell = 256$ bits

Overview of Modular Exponentiations

Algorithm	Line	Computation	PLE		PRE	FBE		GMT
			ℓ	λ	ℓ	ℓ	b	
GenShuffle	1a	$(a_i, b_i) \in \mathcal{G}^2$	–	–	–	–	–	$2N$
	1b	$pk \in \mathcal{G}$	–	–	–	–	–	1
	5	$\tilde{a}_i \leftarrow a_i \cdot pk^{\tilde{r}_i}$	–	–	–	N	pk	–
	6	$\tilde{b}_i \leftarrow b_i \cdot g^{\tilde{r}_i}$	–	–	–	N	g	–
GenProof	4	$c_{j_i} \leftarrow h_i \cdot g^{r_{j_i}}$	–	–	–	N	g	–
	11	$\hat{c}_i \leftarrow g^{\hat{r}_i} \cdot \hat{c}_{i-1}^{\tilde{u}_i}$	–	N	–	N	g	–
	15	$\hat{t}_i \leftarrow g^{\hat{\omega}_i} \cdot \hat{c}_{i-1}^{\tilde{\omega}_i}$	N	–	–	N	g	–
	17	$t_1 \leftarrow g^{\omega_1}$	–	–	–	1	g	–
	18	$t_2 \leftarrow g^{\omega_2}$	–	–	–	1	g	–
	19	$t_3 \leftarrow g^{\omega_3} \cdot \prod_{i=1}^{N} h_i^{\tilde{\omega}_i}$	–	–	N	1	g	–
	20	$t_{4,1} \leftarrow pk^{-\omega_4} \cdot \prod_{i=1}^{N} \tilde{a}_i^{\tilde{\omega}_i}$	–	–	N	1	pk	–
	21	$t_{4,2} \leftarrow g^{-\omega_4} \cdot \prod_{i=1}^{N} \tilde{b}_i^{\tilde{\omega}_i}$	–	–	N	1	g	–
Total			$10N + 5$					$2N + 1$

PLE = plain, PRE = product, FBE = fixed-base, GMT = group membership

Overview of Modular Exponentiations

Algorithm	Line	Computation	PLE		PRE	FBE		GMT
			ℓ	λ	ℓ	ℓ	b	
GenShuffle	1a	$(a_i, b_i) \in \mathcal{G}^2$	–	–	–	–	–	$2N$
	1b	$pk \in \mathcal{G}$	–	–	–	–	–	1
	5	$\tilde{a}_i \leftarrow a_i \cdot pk^{\tilde{r}_i}$	–	–	–	N	pk	–
	6	$\tilde{b}_i \leftarrow b_i \cdot g^{\tilde{r}_i}$	–	–	–	N	g	–
GenProof	4	$c_{j_i} \leftarrow h_i \cdot g^{r_{j_i}}$	–	–	–	N	g	–
	11	$\hat{c}_i \leftarrow g^{\hat{r}_i} \cdot \hat{c}_{i-1}^{\tilde{u}_i}$	–	N	–	N	g	–
	15	$\hat{t}_i \leftarrow g^{\hat{\omega}_i} \cdot \hat{c}_{i-1}^{\tilde{\omega}_i}$	N	–	–	N	g	–
	17	$t_1 \leftarrow g^{\omega_1}$	–	–	–	1	g	–
	18	$t_2 \leftarrow g^{\omega_2}$	–	–	–	1	g	–
	19	$t_3 \leftarrow g^{\omega_3} \cdot \prod_{i=1}^{N} h_i^{\tilde{\omega}_i}$	–	–	N	1	g	–
	20	$t_{4,1} \leftarrow pk^{-\omega_4} \cdot \prod_{i=1}^{N} \tilde{a}_i^{\tilde{\omega}_i}$	–	–	N	1	pk	–
	21	$t_{4,2} \leftarrow g^{-\omega_4} \cdot \prod_{i=1}^{N} \tilde{b}_i^{\tilde{\omega}_i}$	–	–	N	1	g	–
Total			$10N+5$					$2N+1$

PLE = plain, PRE = product, FBE = fixed-base, GMT = group membership

Overview of Modular Exponentiations

Algorithm	Line	Computation	PLE		PRE	FBE		GMT
			ℓ	λ	ℓ	ℓ	b	
GenShuffle	1a	$(a_i, b_i) \in \mathcal{G}^2$	–	–	–	–	–	$2N$
	1b	$pk \in \mathcal{G}$	–	–	–	–	–	1
	5	$\tilde{a}_i \leftarrow a_i \cdot pk^{\tilde{r}_i}$	–	–	–	N	pk	–
	6	$\tilde{b}_i \leftarrow b_i \cdot g^{\tilde{r}_i}$	–	–	–	N	g	–
GenProof	4	$c_{j_i} \leftarrow h_i \cdot g^{r_{j_i}}$	–	–	–	N	g	–
	11	$\hat{c}_i \leftarrow g^{\hat{r}_i} \cdot \hat{c}_{i-1}^{\tilde{u}_i}$	–	N	–	$2N$	g, h	–
	15	$\hat{t}_i \leftarrow g^{\hat{\omega}_i} \cdot \hat{c}_{i-1}^{\tilde{\omega}_i}$	N	–	–	$2N$	g, h	–
	17	$t_1 \leftarrow g^{\omega_1}$	–	–	–	1	g	–
	18	$t_2 \leftarrow g^{\omega_2}$	–	–	–	1	g	–
	19	$t_3 \leftarrow g^{\omega_3} \cdot \prod_{i=1}^{N} h_i^{\tilde{\omega}_i}$	–	–	N	1	g	–
	20	$t_{4,1} \leftarrow pk^{-\omega_4} \cdot \prod_{i=1}^{N} \tilde{a}_i^{\tilde{\omega}_i}$	–	–	N	1	pk	–
	21	$t_{4,2} \leftarrow g^{-\omega_4} \cdot \prod_{i=1}^{N} \tilde{b}_i^{\tilde{\omega}_i}$	–	–	N	1	g	–
Total			$10N + 5$					$2N + 1$

PLE = plain, PRE = product, FBE = fixed-base, GMT = group membership

Improving GenProof

- Line 11 of GenProof:
  - Raising the recursion to the exponent by $R_i = \hat{r}_i + \tilde{u}_i R_{i-1}$ and $U_i = \tilde{u}_i U_{i-1}$ implies

$$\hat{c}_i = g^{\hat{r}_i} \cdot \hat{c}_{i-1}^{\tilde{u}_i} = g^{\hat{r}_i} \cdot (g^{R_{i-1}} \cdot h^{U_{i-1}})^{\tilde{u}_i} = g^{\hat{r}_i + \tilde{u}_i R_{i-1}} \cdot h^{\tilde{u}_i U_{i-1}}$$
$$= g^{R_i} \cdot h^{U_i}$$

  - For $R_0 = 0$ and $U_0 = 1$, we get $\hat{c}_0 = h$ (see Line 8)

- Line 15 of GenProof:

$$\hat{t}_i = g^{\hat{\omega}_i} \cdot \hat{c}_{i-1}^{\tilde{\omega}_i} = g^{\hat{\omega}_i} \cdot (g^{R_{i-1}} \cdot h^{U_{i-1}})^{\tilde{\omega}_i} = g^{\hat{\omega}_i + \tilde{\omega}_i R_{i-1}} \cdot h^{\tilde{\omega}_i U_{i-1}}$$

Overview of Modular Exponentiations

Algorithm	Line	Computation	PLE		PRE		FBE		GMT
			ℓ	λ	ℓ	λ	ℓ	b	
CheckProof	1a	$t \in \mathcal{G} \times \mathcal{G} \times \mathcal{G} \times \mathcal{G}^2 \times \mathcal{G}^N$	–	–	–	–	–	–	$N+5$
	1b	$\boldsymbol{c} \in \mathcal{G}^N$, $\hat{\boldsymbol{c}} \in \mathcal{G}^N$	–	–	–	–	–	–	$2N$
	1c	$(a_i, b_i) \in \mathcal{G}^2$, $(\tilde{a}_i, \tilde{b}_i) \in \mathcal{G}^2$	–	–	–	–	–	–	$4N$
	1d	$pk \in \mathcal{G}$	–	–	–	–	–	–	1
	7	$\hat{c} \leftarrow \hat{c}_N \cdot h^{-u}$	–	–	–	–	1	h	–
	8	$\tilde{c} \leftarrow \prod_{i=1}^N c_i^{u_i}$	–	–	–	N	–	–	–
	9	$\tilde{a} \leftarrow \prod_{i=1}^N a_i^{u_i}$	–	–	–	N	–	–	–
	10	$\tilde{b} \leftarrow \prod_{i=1}^N b_i^{u_i}$	–	–	–	N	–	–	–
	13	$\hat{t}'_i \leftarrow \hat{c}_i^{\,c} \cdot g^{\hat{s}_i} \cdot \hat{c}_{i-1}^{\tilde{s}_i}$	N	N	–	–	N	g	–
	14	$t'_1 \leftarrow \bar{c}^{\,c} \cdot g^{s_1}$	–	1	–	–	1	g	–
	15	$t'_2 \leftarrow \hat{c}^{\,c} \cdot g^{s_2}$	–	1	–	–	1	g	–
	16	$t'_3 \leftarrow \tilde{c}^{\,c} \cdot g^{s_3} \cdot \prod_{i=1}^N h_i^{\tilde{s}_i}$	–	1	N	–	1	g	–
	17	$t'_{4,1} \leftarrow \tilde{a}^c \cdot pk^{-s_4} \cdot \prod_{i=1}^N \tilde{a}_i^{\tilde{s}_i}$	–	1	N	–	1	pk	–
	18	$t'_{4,2} \leftarrow \tilde{b}^c \cdot g^{-s_4} \cdot \prod_{i=1}^N \tilde{b}_i^{\tilde{s}_i}$	–	1	N	–	1	g	–
Total			$9N+11$						$7N+6$

Overview of Modular Exponentiations

Algorithm	Line	Computation	PLE		PRE		FBE		GMT
			ℓ	λ	ℓ	λ	ℓ	b	
CheckProof	1a	$t \in \mathcal{G} \times \mathcal{G} \times \mathcal{G} \times \mathcal{G}^2 \times \mathcal{G}^N$	–	–	–	–	–	–	$N+5$
	1b	$\boldsymbol{c} \in \mathcal{G}^N$, $\hat{\boldsymbol{c}} \in \mathcal{G}^N$	–	–	–	–	–	–	$2N$
	1c	$(a_i, b_i) \in \mathcal{G}^2$, $(\tilde{a}_i, \tilde{b}_i) \in \mathcal{G}^2$	–	–	–	–	–	–	$4N$
	1d	$pk \in \mathcal{G}$	–	–	–	–	–	–	1
	7	$\hat{c} \leftarrow \hat{c}_N \cdot h^{-u}$	–	–	–	–	1	h	–
	8	$\tilde{c} \leftarrow \prod_{i=1}^N c_i^{u_i}$	–	–	–	N	–	–	–
	9	$\tilde{a} \leftarrow \prod_{i=1}^N a_i^{u_i}$	–	–	–	N	–	–	–
	10	$\tilde{b} \leftarrow \prod_{i=1}^N b_i^{u_i}$	–	–	–	N	–	–	–
	13	$\hat{t}'_i \leftarrow \hat{c}_i^{\,c} \cdot g^{\hat{s}_i} \cdot \hat{c}_{i-1}^{\tilde{s}_i}$	N	N	–	–	N	g	–
	14	$t'_1 \leftarrow \bar{c}^{\,c} \cdot g^{s_1}$	–	1	–	–	1	g	–
	15	$t'_2 \leftarrow \hat{c}^{\,c} \cdot g^{s_2}$	–	1	–	–	1	g	–
	16	$t'_3 \leftarrow \tilde{c}^{\,c} \cdot g^{s_3} \cdot \prod_{i=1}^N h_i^{\tilde{s}_i}$	–	1	N	–	1	g	–
	17	$t'_{4,1} \leftarrow \tilde{a}^c \cdot pk^{-s_4} \cdot \prod_{i=1}^N \tilde{a}_i^{\tilde{s}_i}$	–	1	N	–	1	pk	–
	18	$t'_{4,2} \leftarrow \tilde{b}^c \cdot g^{-s_4} \cdot \prod_{i=1}^N \tilde{b}_i^{\tilde{s}_i}$	–	1	N	–	1	g	–
Total			$9N+11$						$7N+6$

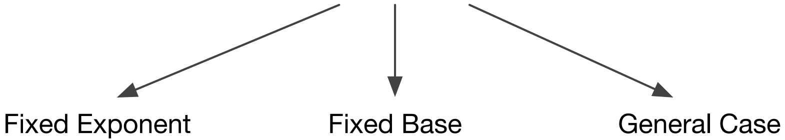
Overview of Modular Exponentiations

Algorithm	Line	Computation	PLE		PRE		FBE		GMT
			ℓ	λ	ℓ	λ	ℓ	b	
CheckProof	1a	$t \in \mathcal{G} \times \mathcal{G} \times \mathcal{G} \times \mathcal{G}^2 \times \mathcal{G}^N$	–	–	–	–	–	–	$N+5$
	1b	$\boldsymbol{c} \in \mathcal{G}^N$, $\hat{\boldsymbol{c}} \in \mathcal{G}^N$	–	–	–	–	–	–	$2N$
	1c	$(a_i, b_i) \in \mathcal{G}^2$, $(\tilde{a}_i, \tilde{b}_i) \in \mathcal{G}^2$	–	–	–	–	–	–	$4N$
	1d	$pk \in \mathcal{G}$	–	–	–	–	–	–	1
	7	$\hat{c} \leftarrow \hat{c}_N \cdot h^{-u}$	–	–	–	–	1	h	–
	8	$\tilde{c} \leftarrow \prod_{i=1}^N c_i^{u_i}$	–	–	–	N	–	–	–
	9	$\tilde{a} \leftarrow \prod_{i=1}^N a_i^{u_i}$	–	–	–	N	–	–	–
	10	$\tilde{b} \leftarrow \prod_{i=1}^N b_i^{u_i}$	–	–	–	N	–	–	–
	13	$\hat{t}'_i \leftarrow \hat{c}_i^{\,c} \cdot g^{\hat{s}_i} \cdot \hat{c}_{i-1}^{\tilde{s}_i}$	N	N	$\boxed{N}$	$\boxed{2N}$	N	–	–
	14	$t'_1 \leftarrow \bar{c}^{\,c} \cdot g^{s_1}$	–	1	–	–	1	g	–
	15	$t'_2 \leftarrow \hat{c}^{\,c} \cdot g^{s_2}$	–	1	–	–	1	g	–
	16	$t'_3 \leftarrow \tilde{c}^{\,c} \cdot g^{s_3} \cdot \prod_{i=1}^N h_i^{\tilde{s}_i}$	–	1	N	–	1	g	–
	17	$t'_{4,1} \leftarrow \tilde{a}^c \cdot pk^{-s_4} \cdot \prod_{i=1}^N \tilde{a}_i^{\tilde{s}_i}$	–	1	N	–	1	pk	–
	18	$t'_{4,2} \leftarrow \tilde{b}^c \cdot g^{-s_4} \cdot \prod_{i=1}^N \tilde{b}_i^{\tilde{s}_i}$	–	1	N	–	1	g	–
Total			$9N+11$						$7N+6$

Improving CheckProof

- The only purpose of the values $\hat{t}'_i$ in Line 13 of CheckProof is to compare them with the given values $\hat{t}_i$ in Line 19
- These tests can be conducted using a mix of batch verification techniques

$$\hat{t}_i \stackrel{?}{=} \hat{c}_i{}^{c} \cdot g^{\hat{s}_i} \cdot \hat{c}_{i-1}^{\tilde{s}_i}$$



Performance Improvement

Partly optimized				Fully optimized				
Generate		Verify		Generate		Verify $s = \lambda$		N
6742	0.21	5406	0.30	3908	0.12	1861	0.10	100
6264	0.20	5233	0.29	3230	0.10	1740	0.10	1 000
5971	0.19	5162	0.29	2817	0.09	1730	0.10	10 000
5782	0.18	5117	0.28	2546	0.08	1729	0.10	100 000
5640	0.18	5083	0.28	2346	0.07	1729	0.10	1 000 000

$\lambda = 128$, $\ell = 3072$ bits

Performance Improvement

Partly optimized				Fully optimized				
Generate		Verify		Generate		Verify $s = \lambda$		N
6742	0.21	5406	0.30	3908	0.12	1861	0.10	100
6264	0.20	5233	0.29	3230	0.10	1740	0.10	1 000
5971	0.19	5162	0.29	2817	0.09	1730	0.10	10 000
5782	0.18	5117	0.28	2546	0.08	1729	0.10	100 000
5640	0.18	5083	0.28	2346	0.07	1729	0.10	1 000 000

$\lambda = 128$, $\ell = 3072$ bits

Performance Improvement

Partly optimized				Fully optimized				
Generate		Verify		Generate		Verify $s = \lambda$		N
6742	0.21	5406	0.30	3908	0.12	1861	0.10	100
6264	0.20	5233	0.29	3230	0.10	1740	0.10	1 000
5971	0.19	5162	0.29	2817	0.09	1730	0.10	10 000
5782	0.18	5117	0.28	2546	0.08	1729	0.10	100 000
5640	0.18	5083	0.28	2346	0.07	1729	0.10	1 000 000

$\lambda = 128$, $\ell = 3072$ bits

Performance Improvement

Partly optimized				Fully optimized				
Generate		Verify		Generate		Verify $s = \lambda$		N
822	0.28	768	0.35	445	0.15	374	0.17	100
763	0.26	742	0.34	362	0.12	356	0.16	1 000
725	0.25	731	0.34	308	0.11	354	0.16	10 000
699	0.24	725	0.33	270	0.09	354	0.16	100 000
683	0.23	721	0.33	248	0.09	354	0.16	1 000 000

$\lambda = 128$, $\ell = 256$ bits

Performance Improvement

Partly optimized				Fully optimized				
Generate		Verify		Generate		Verify $s = \lambda$		N
822	0.28	768	0.35	445	0.15	374	0.17	100
763	0.26	742	0.34	362	0.12	356	0.16	1 000
725	0.25	731	0.34	308	0.11	354	0.16	10 000
699	0.24	725	0.33	270	0.09	354	0.16	100 000
683	0.23	721	0.33	248	0.09	354	0.16	1 000 000

$\lambda = 128$, $\ell = 256$ bits

Performance Improvement

Partly optimized				Fully optimized				
Generate		Verify		Generate		Verify $s = \lambda$		N
822	0.28	768	0.35	445	0.15	374	0.17	100
763	0.26	742	0.34	362	0.12	356	0.16	1 000
725	0.25	731	0.34	308	0.11	354	0.16	10 000
699	0.24	725	0.33	270	0.09	354	0.16	100 000
683	0.23	721	0.33	248	0.09	354	0.16	1 000 000

$\lambda = 128$, $\ell = 256$ bits

Outline

Conclusion

- Product and fixed-base exponentiation algorithms improve the performance by approx. one order of magnitude
- Batch verification algorithms improve the performance by one (general case) respectively two (fixed base/exponent) orders of magnitude
- Using square root witnesses, group membership in $\mathbb{G}_q \subset \mathbb{Z}_p^*$ can be tested at almost no cost
- Applying these techniques to Wikström's shuffle proof improves the overall performance by approx. one order of magnitude