Efficient Implementations of Rainbow and UOV using AVX2

Kyung-Ah Shim (NIMS), Sangyub Lee (NIMS),
Namhun Koo (Ewha Womans University)

NIMS

June 27, 2019

Outline

- Introduction
- UOV
- Profiling UOV Implementation
- Fast Method for Solving Linear Systems
- Precomputation
- Resilience against Leakage or Reuse of Precomputed Values
- Conclusion

MQ-Signature Scheme

- Hard problem: hardness of solving large systems of multivariate quadratic equations, called MQ-problem which is known to be NP-complete.
- Basic structure: a public key is a system of multivariate quadratic polynomials and a trapdoor is hidden in secret affine layers using the ASA (affine-substitution-affine) structure.
- Advanced attacks
  - Key recovery attack on HFEv-, GeMSS
  - MinRank attack and RBS attacks on Rainbow due to its multi-layered structure.
  - The new attacks recovered the secret key at the security level 1 parameter of the second-round submission in 53 hours on a laptop.
  - Rainbow team replaces the security level 1 (resp. 3) parameter with its security level 3 (resp. 5) parameter.

UOV

Main Parameters.

- $\mathbb{F}_q$: the finite field of q elements
- m: the number of polynomials in the public key
- v: the number of Vinegar variables
- o: the number of Oil variables in UOV, $m = o$
- n: the number of variables in the public key, $n = m + v$.

Let v and o be positive integers such that $n = v + o$. Define sets of integers

$$V = \{1, \cdots, v\},\ O = \{v + 1, \cdots, v + o\},$$

$$|V| = v,\ |O| = o,\ m = o,\ n = v + o = v + m.$$

UOV

- A central map $\mathcal{F} : \mathbb{F}_q^n \to \mathbb{F}_q^m$, $\mathcal{F} = (\mathcal{F}^{(1)}, \cdots, \mathcal{F}^{(o)})$ is o multivariate quadratic equations with n variables $x_1, \cdots, x_n$ defined by

$$\mathcal{F}^{(k)}(\mathbf{x}) = \sum_{i \in O, j \in V} \alpha_{ij}^{(k)} x_i x_j + \sum_{i,j \in V, i \leq j} \beta_{ij}^{(k)} x_i x_j + \sum_{i \in V \cup O} \gamma_i^{(k)} x_i + \eta^{(k)}.$$

- An invertible affine map $\mathcal{T} : \mathbb{F}_q^n \to \mathbb{F}_q^n$ is required to destroy the missing Oil$*$Oil structure of $\mathcal{F}$.
- A public key is $\mathcal{P} = \mathcal{F} \circ \mathcal{T}$ that seems to be hardly distinguishable from a random quadratic system, thus be hard to invert.

UOV

- **KeyGen**(1^λ). Given a security parameter λ,
  - a secret key is $SK = (\mathcal{F}, T)$,
  - a public key is $PK = \mathcal{P} = \mathcal{F} \circ \mathcal{T}$.
- **Sign**$(SK, \mathtt{m})$. For a message $\mathtt{m}$and a collision-resistant hash function, $\mathcal{H} : \{0,1\}^* \rightarrow \mathbb{F}_q^m$, do the followings :
  - Choose a λ-bit random number r and compute $\mathbf{h} = \mathcal{H}(\mathtt{m}, r) \in \mathbb{F}_q^m$.
  - Compute $\alpha = \mathcal{F}^{-1}(\mathbf{h})$, i.e. $\mathcal{F}(\mathbf{h}) = \alpha$: First choose $s_V = (s_1, \cdots, s_v) \in \mathbb{F}_q^v$ at random, substitute s_V into o equations $\mathcal{F}^{(k)}$ $(1 \leq k \leq o)$ and get $(s_{v+1}, \cdots, s_{v+o})$ by solving a linear system of o equations with o unknowns $x_{v+1}, \cdots, x_{v+o}$ using the Gaussian elimination. If the linear systems has no solution, choose another vector of Vinegar values s'_V and try again.
  - Calculate $\sigma = \mathcal{T}^{-1}(\alpha)$ and output $\tau = (\sigma, r)$ as signature on $\mathtt{m}$.
- **Verify**$(PK, \mathtt{m}, \sigma)$. For signature on $\mathtt{m}$ $(\tau, \mathtt{m})$ and a public key $\mathcal{P}$, check the equality $\mathcal{P}(\sigma) = h(\mathtt{m}, r)$. If it holds, output valid.

UOV

Algorithm 1 UOV Signature Generation

Require: document d, UOV private key $(\mathcal{F}, InvT)$, length of the salt l.
Ensure: signature $\sigma = (\mathbf{z}, r) \in \mathbb{F}^n \times \{0,1\}^l$ such that $\mathcal{P}(\mathbf{z}) = \mathcal{H}(\mathcal{H}(d)||r)$.

repeat
  $y_1, ..., y_v \leftarrow_R \mathbb{F}$
  $\hat{f}^{(v_1+1)}, ..., \hat{f^{(n)}} \leftarrow f^{(v+1)}(y_1, ..., y_v), ..., f^{(n)}(y, ..., y_v)$
  $(\hat{F}, C_F) \leftarrow \texttt{Aff}^{-1}(\hat{f}^{(v+1)}, ..., \hat{f}^{(n)})$
until IsInvertible($\hat{F}$) $==$ **TRUE**
$InvF = \hat{F}^{-1}$
$r \leftarrow \{0,1\}^l$
$\mathbf{x} \leftarrow \mathcal{H}(\mathcal{H}(d)||r)$
$(y_{v+1}, ..., y_n) \leftarrow InvF \cdot ((x_{v+1}, ..., x_n) - C_F)$
$\mathbf{z} = InvT \cdot \mathbf{y}$
$\sigma \leftarrow (\mathbf{z}, r)$
Return σ =0

Major Computations in UOV and Rainbow

Major Computations in UOV and Rainbow. A main idea to invert a system of quadratic equations in UOV and Rainbow is to convert the quadratic system to a linear system by substituting random Vinegar values into the Vinegar variables of the central quadratic polynomials.

- **Substitution of Vinegar Values into the Central Polynomials.** Calculations for substituting random Vinegar values into the central polynomials are required. Since there are a large number of quadratic terms with Vinegar×Vinegar indexes and Vinegar×Oil indexes being substituted by the Vinegar values, the computations are heavy.
- **Solving Linear System.** Solving the linear systems after the Vinegar value substitution are required. Gaussian elimination is used to find a solution of the linear system, whose complexity is $O(k^3)$ for a $k \times k$ random matrix.

Suggested parameters of UOV/Rainbow

- Intersection attacks

Scheme	Security Category	I	III	V
UOV	λ (Gates)	146	212	274
	(o, v)	(46, 70)	(72, 109)	(96, 144)
	Direct Attack	144.05	212.05	274.847
	Intersection Attack	166.87	236.36	291.501
Rainbow	λ (Gates)	177	226	-
	(v, o_1, o_2)	(68, 32, 48)	(96, 36, 64)	-
	Direct Attack	234	285	-
	Intersection Attack	177	226	-

Table: Suggested Parameters of UOV/Rainbow at Three SCs.

UOV/Rainbow Implementation Results

- An equivalent key of UOV of the form $\mathcal{T} = \begin{pmatrix} I & T' \\ 0 & I \end{pmatrix}$ and a linear map.
- Intel(R) Core(TM) i9-10900X CPU running at the constant clock frequency of 3.70GHz.
- Each result is an average of 10,000 measurements for each function using the C programming language with GNU GCC version 10.1.0 compiler on Centos 7.9.2009. Hyperthreading and Turbo Boost are switched off.

Scheme	SC	I	III	V
UOV	KeyGen.	29,077,126	98,870,925	161,016,435
	Sign	201,834	707,959	1,486,775
	Verify	125,312	222,012	485,344
Rainbow	KeyGen.	65,099,975	214,977,689	—
	Sign	322,799	807,309	—
	Verify	151,466	395,259	—

Profiling UOV/Rainbow Implementation.

- Run-time of UOV signing is mostly dominated by the two operations.
- In UOV, the proportion of cycles spent in the Vinegar values substitutions and in finding an inverse matrix are up to 52 % and 47 %, respectively.

Scheme	Layer	Operations	I	III	V
UOV	1	Vinegar Value Substitutions	58.09 %	48.37 %	56.60 %
		Computation of LS_V^{-1}	36.38 %	47.98 %	41.71 %
		Etc.	5.53 %	3.65 %	1.69 %
Rainbow	1	Vinegar Value Substitutions	18.58 %	34.37 %	—
		Computation of $LS_{V,1}^{-1}$	11.37 %	8.03 %	—
		Etc.	1.26 %	0.68 %	—
	2	Vinegar Value Substitutions	39.54 %	29.07 %	—
		Computation of $LS_{V,2}^{-1}$	25.38 %	25.34 %	—
		Etc.	3.88 %	2.51 %	—

Strategy

- To accelerate solving linear systems, we will reduce the sizes of matrices being inverted by half based on a block matrix inversion method.
- We will use precomputation to handle both of the Vinegar value substitution and solving linear systems.

Block Matric Inversion

Theorem 1. Let R be a matrix partitioned into 2 $\times$ 2 blocks.

(i) Assume A is nonsingular: then the matrix R is invertible if and only if the Schur complement $(D - CA^{-1}B)$ of A is invertible and

$$R^{-1} =$$

$$\begin{pmatrix} A^{-1} + A^{-1}B(D-CA^{-1}B)^{-1}CA^{-1} & -A^{-1}B(D-CA^{-1}B)^{-1} \\ -(D-CA^{-1}B)^{-1}CA^{-1} & (D-CA^{-1}B)^{-1} \end{pmatrix}.$$

(ii) Assume D is nonsingular: then the matrix R is invertible if and only if the Schur complement $(A - BD^{-1}C)$ is invertible and

$$R^{-1} =$$

$$\begin{pmatrix} (A-BD^{-1}C)^{-1} & -(A-BD^{-1}C)^{-1}BD^{-1} \\ -D^{-1}C(A-BD^{-1}C)^{-1} & D^{-1} + D^{-1}C(A-BD^{-1}C)^{-1}BD^{-1} \end{pmatrix}.$$

$\Rightarrow$ It requires two inversions and six matrix multiplications of the half-sized matrices.

Block Matric Inversion

Theorem 2. For a nonsingular $k \times k$ matrix R in the above, $R^{-1} \cdot \alpha$ requires two inversions, two matrix multiplications of the half-sized block matrices and four block matrix-vector products, where k is even and $\alpha = (\alpha_1, \cdots, \alpha_{k/2})^{\mathtt{T}}$.

Proof. A nonsingular square matrix R of 2×2 blocks is represented by the LDU decomposition of block matrices based on the Schur complement as

$$R = \begin{pmatrix} A & B \\ C & D \end{pmatrix} = \begin{pmatrix} I & O \\ CA^{-1} & I \end{pmatrix} \begin{pmatrix} A & O \\ 0 & D - CA^{-1}B \end{pmatrix} \begin{pmatrix} I & A^{-1}B \\ 0 & I \end{pmatrix}$$

$$= L \cdot D_{Sc} \cdot U.$$

$$R^{-1} = U^{-1} \cdot D_{Sc}^{-1} \cdot L^{-1}$$

$$= \begin{pmatrix} I & -A^{-1}B \\ 0 & I \end{pmatrix} \begin{pmatrix} A^{-1} & O \\ 0 & [D - CA^{-1}B]^{-1} \end{pmatrix} \begin{pmatrix} I & 0 \\ -CA^{-1} & I \end{pmatrix}.$$

Block Matric Inversion

$$R^{-1} \cdot \begin{pmatrix} \alpha_1 \\ \cdots \\ \alpha_k \end{pmatrix}$$

$$= \begin{pmatrix} I & -A^{-1}B \\ 0 & I \end{pmatrix} \begin{pmatrix} A^{-1} & O \\ 0 & [D - CA^{-1}B]^{-1} \end{pmatrix} \begin{pmatrix} I & 0 \\ -CA^{-1} & I \end{pmatrix} \begin{pmatrix} \alpha_1 \\ \cdots \\ \alpha_k \end{pmatrix}.$$

Thus, computing $R^{-1} \cdot \alpha$ requires two block matrix multiplications for computing $A^{-1}B$ and $C(A^{-1}B)$, two inversions for A^{-1} and $[D - CA^{-1}B]^{-1}$, and four block matrix-vector products. Its complexity is reduced to $O((k/2)^3)$. Consequently, we reduce from six block matrix multiplications to two block matrix multiplications and four block matrix-vector products. $\square$

Block Matric Inversion

Repeated BMI: Determine the Minimum Size of a Matrix being Inverted. We want to determine the minimum size of a matrix being inverted.

- After performing the BMI, we reduced the size of a matrix being inverted by half, while the number of inversions for the half-sized matrices increased to two. We can apply the BMI again to these two half-sized matrices which results in four inversions of $k/4 \times k/4$ matrices and extra operations.
- Like this, for $m = 2^l \cdot m'$, we can apply the BMI l times. We define the number of these iterations of the BMI as a depth. We cannot expect that l iterations will always be effective, because 2^l inversions of $k/2^l \times k/2^l$ matrices are required.

Block Matric Inversion

- The matrix sizes should be multiples of 2 and multiples of 4, respectively. If $m = 46$, then we cannot use the BMI with the depth 2 since it is not a multiple of 4. After obtaining LS_V from the Vinegar value substitution, we set $LS_V = \begin{pmatrix} A & B \\ C & D \end{pmatrix}$, if A or $[D - CA^{-1}B]$ is not invertible then we choose another Vinegar values. The probability that the matrices are invertible is 99.223%.
- Compared to Gaussian elimination, the larger the size, the greater the performance improvement and the higher the security category, the greater the effect.
  - By using BMI with the depth 1, we get speedups of 23.1%, 41%, 64.6%, and 50.9% at the sizes, 46, 72, 96, and 100, respectively. Especially excellent improvements of 61.3% and 64.6% in the case of 64 and 96, respectively, are due to the fact that the multiples of 32 are optimal parameters which are suitable for the AVX2 vectorization.
  - In the BMI with the depth 2, their speedups are 46.54%, 68.23%, and 57.58% at the sizes, 72, 96, and 100, respectively.

Block Matric Inversion

Matrix Size	GE	BMI (Depth 1)	BMI (Depth 2)
46	94,033	72,302	—
48	101,498	75,136	72,479
50	142,243	82,768	—
56	175,195	103,357	99,445
64	225,081	87,150	70,091
68	322,315	187,947	170,788
72	355,173	208,355	190,480
96	713,462	252,538	226,627
100	923,489	453,441	391,747

Table: Gaussian Elimination (GE) vs. Block Matrix Inversion Technique in CPU Cycles.

Block Matric Inversion

- Compared to UOV implemented with Gaussian elimination, by using the BMI with the depth 1, we obtain speedups of 12.36%, 20.41%, and 32.42% at the three security categories, respectively.

Scheme	SC	I	III	V
UOV	G.E.	201,834	707,959	1,486,775
	BMI (Depth 1)	176,884	563,519	1,004,704
	BMI (Depth 2)	—	535,660	981,351
Rainbow	G.E.	322,799	807,309	—
	BMI (Depth 1)	270,731	650,400	—
	BMI (Depth 2)	271,986	639,965	—

Table: UOV Implementation Results on the Intel at Three SCs, in CPU cycles.

Precomputation

Offline Signing of UOV.

- After choosing random Vinegar values $s_V = (s_1, \cdots, s_v) \in \mathbb{F}_q^v$, substitute s_V into o equations $\mathcal{F}^{(k)}$ $(1 \leq k \leq o)$ to get the linear system LS_V of o equations and o unknowns and a constant vector $c_V = (c_1, \cdots, c_m)$.
- Compute LS_V^{-1}. If LS_V is not invertible then go back to the first step.
- Store $< s_V, c_V, LS_V^{-1} >$ as the precomputed values.

Online Signing of UOV.

- Choose a random salt r and compute $h = \mathcal{H}(\mathcal{H}(\mathtt{m})||r)$ for a message $\mathtt{m}$.
- From $< s_V = (s_1, \cdots, s_V)$, $c_V = (c_1, \cdots, c_m)$, $LS_V^{-1} >$, compute $LS_V^{-1} \cdot h_V^{\mathtt{T}} = \alpha$, where $h_V = (h_1 - c_1, \cdots, h_m - c_m)$ and $h = (h_1, \cdots, h_m)$.
- Compute $T^{-1} \cdot (S_V, \alpha)^{\mathtt{T}} = \sigma$ and output $\tau = (\sigma, r)$ as a signature on $\mathtt{m}$.

Precomputation

Scheme	Security Category	Unit	I	III	V
UOV	Sign w/o Precomp.	cycle	201 834	707 959	1 486 775
	Precomp. (offline)	cycle	189 224	690 586	1 460 168
	Sign w/ Precomp. (online)		11 788	19 439	23 133
	Total (offline + online)		201 012	710 025	1 483 301
	Precomp. Memory Cost per Sig.	byte	2 256	5 402	9 504
Rainbow	Sign w/o Precomp.	cycle	68 203	322 799	807 309
	Precomp. (offline)	cycle	37 212	173 204	508 890
	Sign w/ Precomp. (online)		31 973	142 179	278 511
	Total (offline + online)		69 185	315 383	787 401
	Precomp. Memory Cost per Sig.	byte	2 152	4 792	5 648

Leakage of Precomputed Values

Store $< s_V, c_V, LS_V^{-1} >$ **Securely.** The precomputed values $< s_V, c_V, LS_V^{-1} >$ should be stored securely. If some precomputed values together with signatures generated by them are exposed then the secret key of UOV is completely recovered.

Theorem 3. If $(n+1)$ tuples $< \mathtt{m}^{(i)}, \tau^{(i)}, s_V^{(i)},\ c_V^{(i)}, LS_V^{(i)-1} >$ are given such that the $n \times n$ matrix $(\sigma^{(1)\mathtt{T}}\ \sigma^{(2)\mathtt{T}}\ \cdots\ \sigma^{(n)\mathtt{T}})$ is invertible then the secret key of UOV is completely recovered in polynomial-time.

Theorem 4. If $(n+1)$ tuples $< \mathtt{m}^{(i)}, \tau^{(i)}, s_V^{(i)},\ c_V^{(i)}, (LS_V^{(i)})^{-1} >$ are given such that the $n \times n$ matrix $(\sigma^{(1)\mathtt{T}}\ \sigma^{(2)\mathtt{T}}\ \cdots\ \sigma^{(n)\mathtt{T}})$ is invertible then an equivalent key of Rainbow is completely recovered in polynomial-time.

Reuse of Precomputed Values

Do not Reuse $< s_V, c_V, LS_V^{-1} >$. The precomputed value $< s_V, c_V, LS_V^{-1} >$ should not be reused in signing.

Theorem 5. If $(m+1)$ signatures generated by the reused Vinegar values are given then

- the equivalent key of UOV is completely recovered in polynomial time,
- the complexity of the KRAs using good keys on Rainbow is determined by solving a multivariate system of m quadratic equations with o_1 variables.

Now, we provide a more improved analysis: (o_2+1) signatures generated by the fixed Vinegar values lead to the full secret key recovery of Rainbow in polynomial-time.

Theorem 6. If (o_2+1) signatures generated by reusing the precomputed values then an equivalent key of Rainbow is recovered in polynomial-time with high probability.

Remarks

Zambonin *et al.*'s **Variants of Rainbow.** At Africacrypt 2019, Zambonin *et al.* [?] proposed three variants of Rainbow to shorten the secret key size by using fixed Vinegar values and central polynomials substituted by the fixed Vinegar values in key generations. In other words, the secret key includes the fixed Vinegar values and all the signatures are generated by the same Vinegar values in their scheme. Theorem 6 shows that their three variants are entirely broken by the key recovery attacks using good keys.

Applicability of Our Optimization to Cyclic/Compressed Versions of UOV and Rainbow. The signing processes of Cyclic versions of UOV and Rainbow are identical to those of the original versions. The signing processes of their Compressed versions are identical to those of the original versions except that the Compressed versions require additional secret key recoveries before generating a signature. Thus, our BMI method and precomputation can be applied to signing in Cyclic/Compressed versions of UOV and Rainbow.

Comparison

Scheme			UOV	Rainbow
Structure			ASA, Single Layer	ASA, Two Layers
Public Key			$\mathcal{P} = \mathcal{F} \circ \mathcal{T}$	$\mathcal{P} = \mathcal{S} \circ \mathcal{F} \circ \mathcal{T}$
Hard Problems			MQ, EIP	MQ, EIP, MinRank
Invert			Oil-Vinegar Method	Oil-Vinegar Method
Solving Linear Systems	Sign (GE)	I	201 834 cycles	68 203 cycles
		III	707 959 cycles	322 799 cycles
		V	1 486 775 cycles	807 309 cycles
	Sign (BMI)	I	176 884 cycles	—
		III	563 519 cycles	270 731 cycles
		V	981 351 cycles	650 400 cycles
Precomp.	PV		(s_V, c_V, LS_V^{-1})	$(s_{V,1}, c_{V,1}, LS_{V,1}^{-1}, \{\mathcal{F}^{(o_1+i)}(s_V)\}_{i=1}^{o_2})$
	Online Comp.		Two M-V prod.	Subst. of o_1-values, One Inv., Three M-V prod.
	Mem. Cost		$2v + m^2$ bytes	$2v + o_1^2 + (o_2+1)o_2$ bytes
	Sign w/ Precomp.	I	11 968 cycles	32 129 cycles
		III	19 968 cycles	144 735 cycles
		V	23 667 cycles	288 008 cycles
Resistant	PV Leakage		Insecure $(n+1)$	Insecure $(n+1)$
	PV Reuse		Insecure $(m+1)$	Insecure (o_2+1)

Conclusion

- Solving Linear System
  - Block Matrix Inversion: 10-40% faster than Gaussian elimination
- Substitution Vinegar values + Solving Linear system
  - Precomputation: 1/17, 1/36, 1/64 improvements

Thank you.