SMAUG Update to v2.0

Jung Hee Cheon[1, 2], Hyeongmin Choe[1], Dongyeon Hong[2], Jeongdae Hong[3], **Hyoeun Seong**[2], Junbum Shin[2], MinJune Yi[1, 2]

[1]Seoul National University, [2]CryptoLab Inc., [3]Ministry of National Defense

7-th KpqC workshop
November 13, 2023

SMAUG
HEAAN
CRYPTO LAB

Main changes

Through a few improvements, SMAUG has been further enhanced

- More efficient!
  - FO-transform updated to avoid hashing and simplify proof
  - Reference code was partially optimized
- Constant time implementation
  - HWT sampling updated to avoid rejection sampling
  - Conditional statement was removed from ciphertext comparison
- Enhancing code security
  - A bug was fixed in convToIdx
  - Constant time & memory leakage were checked using Valgrind
    - followed the same method as in KPQClean
    - no vulnerabilities were found in the updated version

Main changes

Through a few improvements, SMAUG has been further enhanced

- More efficient!
  - FO-transform updated to avoid hashing and simplify proof
  - Reference code was partially optimized
- Constant time implementation
  - HWT sampling updated to avoid rejection sampling
  - Conditional statement was removed from ciphertext comparison
- Enhancing code security
  - A bug was fixed in convToIdx
  - Constant time & memory leakage were checked using Valgrind
    - followed the same method as in KPQClean
    - no vulnerabilities were found in the updated version

Main changes

Through a few improvements, SMAUG has been further enhanced

- More efficient!
  - FO-transform updated to avoid hashing and simplify proof
  - Reference code was partially optimized
- Constant time implementation
  - HWT sampling updated to avoid rejection sampling
  - Conditional statement was removed from ciphertext comparison
- Enhancing code security
  - A bug was fixed in convToIdx
  - Constant time & memory leakage were checked using Valgrind
    - followed the same method as in KPQClean
    - no vulnerabilities were found in the updated version

Main changes: FO-transform

Tweaked FO-transform generates shared key without ciphertext contribution [GMP21][FIP23]

(a) SMAUG v0.9

Encap(pk):

1: $\mu \leftarrow \{0,1\}^{256}$
2: $\mathsf{seed} \leftarrow G(\mu, H(\mathsf{pk}))$
3: $\mathsf{ct} \leftarrow \mathsf{PKE.Enc}(\mathsf{pk}, \mu; \mathsf{seed})$
4: $K \leftarrow KDF(\mu, H(\mathsf{ct}))$
5: **return** ct, K

Decap(sk, ct): ▷ $\mathsf{sk} = (\mathsf{sk}', d)$

1: $\mu' = \mathsf{SMAUG.PKE.Dec}(\mathsf{sk}', \mathsf{ct})$
2: $\mathsf{seed}' \leftarrow G(\mu', H(\mathsf{pk}))$
3: $\mathsf{ct}' = \mathsf{PKE.Enc}\left(\mathsf{pk}, \mu'; \mathsf{seed}'\right)$
4: **if** $\mathsf{ct} \neq \mathsf{ct}'$ **then**
5: $\quad K' \leftarrow G(d, H(\mathsf{ct}))$
6: **else**
7: $\quad K' \leftarrow G(\mu', H(\mathsf{ct}))$
8: **end if**
9: $\quad$ **return** K'

(b) SMAUG v1.0 (May, 2023)

Encap(pk):

1: $\mu \leftarrow \{0,1\}^{256}$
2: $(K, \mathsf{seed}) \leftarrow G(\mu, H(\mathsf{pk}))$
3: $\mathsf{ct} \leftarrow \mathsf{PKE.Enc}(\mathsf{pk}, \mu; \mathsf{seed})$
4: **return** ct, K

Decap(sk, ct): ▷ $\mathsf{sk} = (\mathsf{sk}', d)$

1: $\mu' = \mathsf{SMAUG.PKE.Dec}(\mathsf{sk}', \mathsf{ct})$
2: $(K', \mathsf{seed}') \leftarrow G(\mu', H(\mathsf{pk}))$
3: $\mathsf{ct}' = \mathsf{PKE.Enc}\left(\mathsf{pk}, \mu'; \mathsf{seed}'\right)$
4: $(\hat{K}, \cdot) \leftarrow G(d, H(\mathsf{ct}))$
5: **if** $\mathsf{ct} \neq \mathsf{ct}'$ **then**
6: $\quad K' \leftarrow \hat{K}$
7: **end if**
8: $\quad$ **return** K'

Main changes: HWT sampling

Hybrid combination of SampleInBall & CWW sampling [Sen21]

(a) rejection sampling used

$\underline{\text{HWT}_h}$:

1: Initialize $\mathbf{c} = c_0 c_1 ... c_{n-1} = 00...0$
2: **for** i from $n-h$ to $n-1$ **do**
3: **while** $(j > i)$ **do** $\triangleright\ j \leq i$
4: $j \xleftarrow{\$} \{0, 1, ..., n-1\}$
5: **end while**
6: $b \leftarrow \{0, 1\}$
7: $c_i := c_j$
8: $c_j := (-1)^b$
9: **end for**
10: **return** $\mathbf{c}$

- Re-seeding is required in some cases
- Number of XOF calls depends on input seed

(b) CWW sampling applied

$\underline{\text{HWT}_h}$:

1: Initialize $\mathbf{c} = c_0 c_1 ... c_{n-1} = 00...0$
2: **for** i from $n-h$ to $n-1$ **do**
3: $s_i \xleftarrow{\$} \{0, 1, ..., 2^{32}-1\}$
4: $j \leftarrow \lfloor (i+1) \cdot s_i / 2^{32} \rfloor$ $\triangleright\ j \leq i$
5: $b \leftarrow \{0, 1\}$
6: $c_i := c_j$
7: $c_j := (-1)^b$
8: **end for**
9: **return** $\mathbf{c}$

- No additinal XOF call required
- Running time doesn't depend on input seed

Benchmark

Performance (cpu cycles)

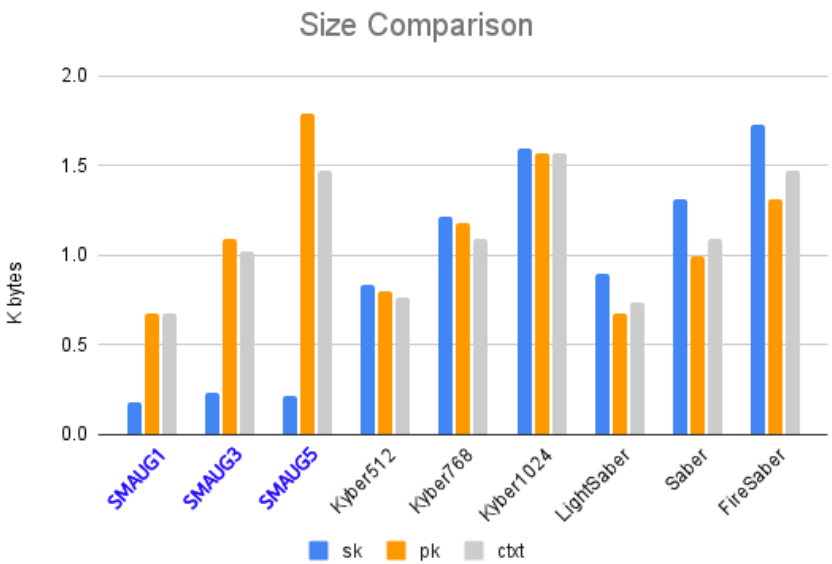
- Intel(R) Core i7-10700K (3.80GHz)
- The compiler gcc 9.4.0 with -O3 and -fomit-frame-pointer.

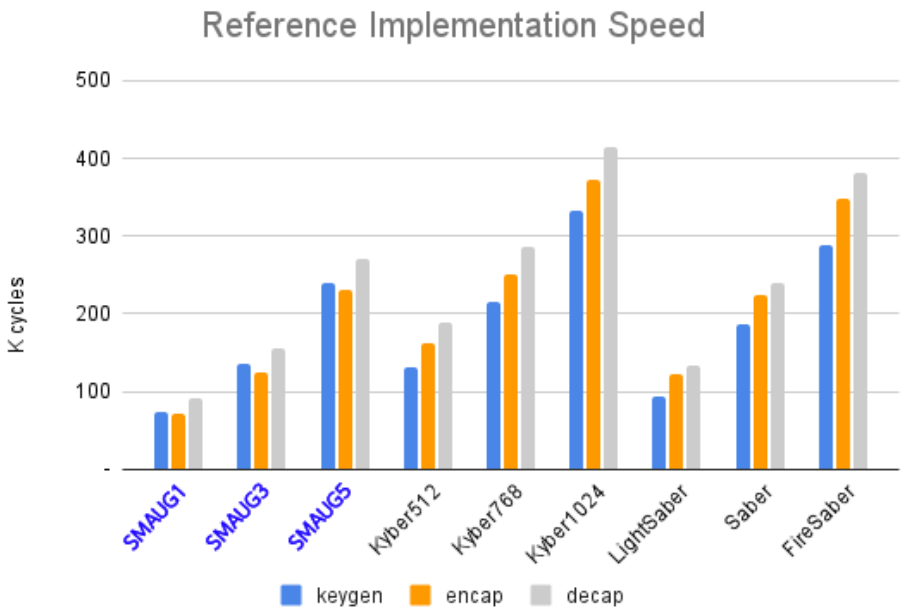
	Smaug128	Smaug192	Smaug256
Keygen	73,061	135,899	240,254
Encap	71,669	123,763	230,521
Decap	91,007	156,293	270,437
Secret key	176 (848)	236 (1324)	218 (2010)
Public key	672	1088	1792
Ciphertext	672	1024	1472

- SMAUG is now about 5-15% faster than previous version
  - Number of XOF calls reduced in HWT sampling
  - Other functions are partly optimizaed in reference code

Benchmark

SMAUG is **efficient & fast!**





- Kyber: https://github.com/pq-crystals/kyber (a621b8d), Saber: https://github.com/KULeuven-COSIC/SABER (f7f39e4)

Conclusion

Reminder:

- It has the advantages of both LWE's security and LWR's efficiency
- Small bandwidth & memory consumption which is suitable to IoT environment

Updated:

- SMAUG is now more fast and securely implemented
- SMAUG offers **superior performance and bandwidth** compared to other lattice schemes that don't use ECC

Conclusion

Reminder:

- It has the advantages of both LWE's security and LWR's efficiency
- Small bandwidth & memory consumption which is suitable to IoT environment

Updated:

- SMAUG is now more fast and securely implemented
- SMAUG offers **superior performance and bandwidth** compared to other lattice schemes that don't use ECC

Thanks 

References I

[FIP23] Federal Information Processing Standards (Draft): FIPS203.
Module-lattice-based key-encapsulation mechanism standard, 2023.
https://csrc.nist.gov/pubs/fips/203/ipd.

[GMP21] Paul Grubbs, Varun Maram, and Kenneth G. Paterson.
Anonymous, robust post-quantum public key encryption.
Cryptology ePrint Archive, Paper 2021/708, 2021.
https://eprint.iacr.org/2021/708.

[Sen21] Nicolas Sendrier.
Secure sampling of constant-weight words – application to bike.
Cryptology ePrint Archive, Paper 2021/1631, 2021.
https://eprint.iacr.org/2021/1631.