Compressed SIKE Round 3 on Cortex-M4

Mila Anastasova[1], Mojtaba Bisheh-Niasar[1], Reza Azarderakhsh[1,2] and Mehran Mozaffari Kermani[3]

[1]CEECS Department and I-SENSE at Florida Atlantic University, Boca Raton, FL, USA
(manastasova2017, mbishehniasa2019, razarderakhsh)@fau.edu

[2]PQSecure Technologies, LLC, Boca Raton, FL, USA

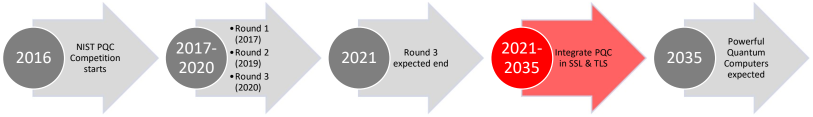
[3]CES Department at University of South Florida, Tampa, FL, USA,
mehran2@usf.edu

August 2021

Content

Introduction

- Quantum computers threaten to break the classical cryptography schemes, such as RSA and the ECC family.
- Led by the increasing capabilities of quantum computers NIST [6] initiated a standardization process.



The Supersingular Isogeny Key Encapsulation

Public Parameters: $p = 2^{e_A}3^{e_B} - 1$, $E_0/\mathbb{F}_{p^2}$, $\{P_A, Q_A\}$ and $\{P_B, Q_B\}$

Alice

Input: -
Output: s, sk_A, pk_A
1. $sk_A \in_R \mathbb{Z}/2^{e_A}\mathbb{Z}$
2. $\phi_A : E_0 \to E_A$ with $ker(\phi_A) = \langle P_A + [sk_A]Q_A \rangle$
3. $pk_A = (E_A, \phi_A(P_B), \phi_A(Q_B))$
4. $s \in_R \{0,1\}^t$

Compress public key

Bob

$\xrightarrow{pk_A}$

Input: pk_A
Output: c, ss

Decompress public key
1. $m \in_R \{0,1\}^t$
2. $r = H(m||pk_A) mod 3^{e_B}$
3. $\phi_B : E_0 \to E_B$ with $ker(\phi_B) = \langle P_B + [r]Q_B \rangle$
4. $pk_B = \{E_B, \phi_B(P_A), \phi_B(Q_A)\}$
5. $\phi'_B : E_A \to E_{AB}$ with $ker(\phi'_B) = \langle \phi_A(P_B) + [r]\phi_A(Q_B) \rangle$
6. $c = (c_0, c_1) = (pk_B, K(j(E_{AB})) \oplus m)$
7. $ss = (J(m||c))$

Compress ciphertext

$\xleftarrow{c}$

Alice

Input: s, sk_B, pk_B, c
Output: ss

Decompress ciphertext
1. $\phi'_A : E_B \to E_{BA}$ with $ker(\phi'_A) = \langle \phi_B(P_A) + [sk_A]\phi_B(Q_A) \rangle$
2. $m' = c_1 \oplus K(j(E_{BA}))$
3. $r' = H(m'||pk_A) mod 3^{e_B}$
4. $\phi''_A : E_0 \to E_{B'}$ with $ker(\phi''_A) = \langle P_B + [r']Q_B \rangle$
5. $pk'_B = \{E_{B'}, \phi''_A(P_A), \phi''_A(Q_A)\}$
6. IF $pk'_B = pk_B$
   $ss = (J(m'||c))$
   ELSE $ss = (J(s||c))$

- Based on secret isogeny maps between supersingular elliptic curves.
- Insignificant communication latency - smallest key sizes among the PQ candidates.
- Large execution time due to sophisticated computations.
- Forms part of the alternate candidates in Round 3 of the NIST standardization process.

Compressed SIKE [2]

- Both parties represent the public key information based on canonical basis.
- The basis generation, paring and discrete logarithms computation add significant overhead to the execution time.
- The insignificant communication latency motivates continuous work on the compressed SIKE protocol [3], [5], [7].

Public Key (De)Compression

1. Compress Public Key

$\langle R_1, R_2 \rangle = E_A\,[3^{e_B}]$
$c_0, c_1 : \phi_A(P_B) =$
$[c_0]\hat{\phi}_A(R_1) + [c_1]\hat{\phi}_A(R_2)$
$d_0, d_1 : \phi_A(Q_B) =$
$[d_0]\hat{\phi}_A(R_1) + [d_1]\hat{\phi}_A(R_2)$
IF $d_1 mod 3^{e_B} = 0$
$\alpha = -d_0^{-1}d_1,\ \beta = c_1 d_0^{-1},$
$\gamma = -c_0 d_0^{-1}$
ELSE
$\alpha = -d_0 d_1^{-1},$
$\beta = -c_1 d_1^{-1}, \gamma = c_0 d_1^{-1}$
$pk_A = pk\_comp_A =$
$\{\alpha, \beta, \gamma, \overline{A}\}$

$\xrightarrow{pk_A}$

2. Decompress Public Key

$\langle R_1, R_2 \rangle = E_A\,[3^{e_B}]$
IF $d_1 mod 3^{e_B} = 0$
$ker(\phi'_B) = \langle R_1 + [((\alpha +$
$[r]\gamma)(1 + [r]\beta))^{-1}]R_2 \rangle$
ELSE
$ker(\phi'_B) = \langle R_1 + [((\alpha +$
$[r]\beta)(1 + [r]\gamma))^{-1}]R_2 \rangle$

Ciphertext (De)Compression

4. Decompress Ciphertext

$\langle S_1, S_2 \rangle \in E_B\,[2^{e_A}]$
$\phi'_A : E_B \to E_{BA}$
with$ker(\phi'_A) = \langle S_1 + [(sk_A a_1 +$
$a_0)^{-1}(sk_A b_1 + b_0)]S_2$
or
$ker(\phi'_A) = \langle S_1 + [(sk_A b_1 +$
$b_0)^{-1}(sk_A a_1 + a_0)]S_2 \rangle$

$\xleftarrow{c}$

3. Compress Ciphertext

$\langle S_1, S_2 \rangle \in E_B\,[2^{e_A}]$
$a_0, b_0 : \phi_B(P_A) =$
$[a_0]\hat{\phi}_B(S_1) + [b_0]\hat{\phi}_B(S_2)$
$a_1, b_1 : \phi_B(Q_A) =$
$[a_1]\hat{\phi}_B(S_1) + [b_1]\hat{\phi}_B(S_2)$
$c_0 = (A, (a_0, b_0, a_1, b_1))$

ARM Cortex-M4

NIST recommended microcontroller for benchmarking.

Features

ARMv7-M architecture

16 32-bit core registers

32 32-bit FP registers

1 CC per instruction except memory accesses

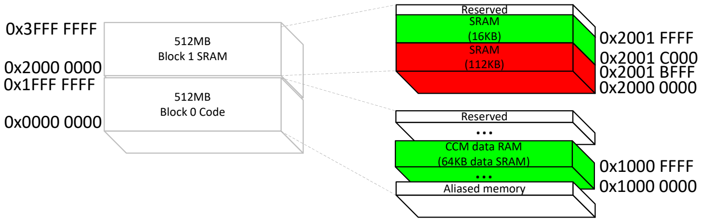
Implementation strategies

Use the entire register set.

Operate on larger operand sets.

Re-organize the instruction flow for efficient design.

ARM Cortex M4 - Memory Map



- Features 1MB of flash and 192KB of RAM - 128KB SRAM and 64KB CCM RAM.
- The 128KB of SRAM - not enough to run compressed SIKEp610.
- We reserved a region inside the CCM RAM to place part of the large data structures, residing into the stack.

Modular subtraction

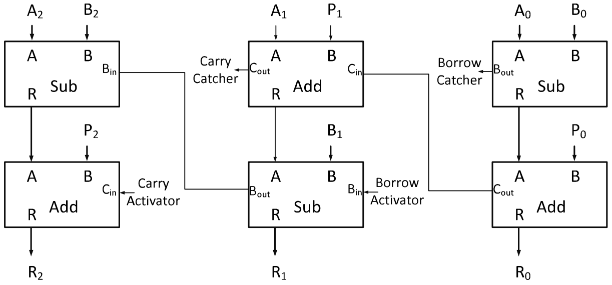
a - b + P

Proposed strategies :

Increase the size of the addition blocks.

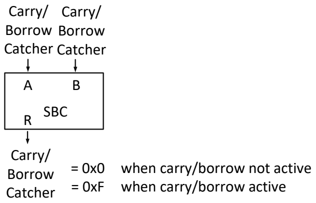
Alternate the add/sub blocks.

Use reduced instruction set for the carry/borrow catcher/activator.

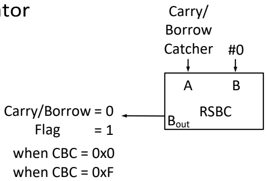


Modular addition

Carry/Borrow Catcher



Carry/Borrow Activator



`a + b - P`

Proposed strategies :

Reduced instruction set for carry/borrow catcher/activator.

Use `SBC` Subtract with Carry flag for Carry/Borrow Catcher.

Use `RSBC` Reverse Subtract with Carry flag for Carry/Borrow Activator. [1]

Multi-Precision Multiplication

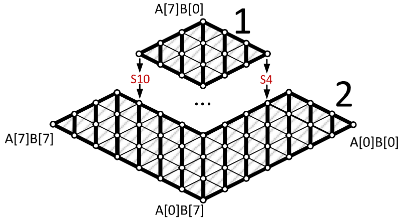
`a * b = (a`$_{n-1}$`, ..., a`$_0$`) * (b`$_{n-1}$`, ..., b`$_0$`) = (c`$_{2n-1}$`, ..., c`$_0$`)`

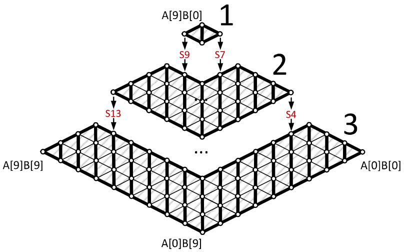
Proposed strategies :

Use the FP register set as L1 cache.

Use the 1 clock cycle instruction `VMOV` to transfer data between `R` and `S` registers [1].

Apply Refined Operand Caching [9].





Results subroutine implementation

Implementation	Timing [cc×10^6]								
	sub_2p	sub_4p	mul	sub_2p	sub_4p	mul	sub_2p	sub_4p	mul
	SIKEp434 compressed			SIKEp503 compressed			SIKEp610 compressed		
SIDH v3.3[1]	577	579	4,940	656	656	4,940	813	813	8,034
This work	135	135	202	147	147	202	180	180	296

- Subtraction improved by around 77% for all security levels.
- Multi-precision multiplication improved by around 96% for all security.

Refer to : [1] [8]

Timing Results @24MHz and @168MHz

Implementation	Timing [cc×10^6] @24MHz				Speedup	Timing Total @168MHz	
	KeyGen	Encaps	Decaps	Total	[%] E+D	[cc×10^6]	[sec]
SIKEp434 compressed							
SIDH v3.3[1]	1,088	1,715	1,272	2,987	94.1	3,017	18.00
Seo et al.[2]	79	133	98	231	24.9	240	1.43
Anastasova et al.[3]	76	119	89	208	16.7	246	1.46
This work	68	99	74	173	-	212	1.26
SIKEp503 compressed							
SIDH v3.3[1]	1,638	2,601	1,920	4,521	94.5	4,519	27.00
Seo et al.[2]	111	181	137	318	21.8	333	1.98
Anastasova et al.[3]	99	164	125	289	13.9	359	2.14
This work	89	143	106	249	-	318	1.89
SIKEp610 compressed							
SIDH v3.3[1]	3,244	4,909	3,889	8,798	94.5	8,775	52.00
Seo et al.[2]	220	333	278	611	20.4	627	3.73
Anastasova et al.[3]	191	306	255	561	13.2	635	3.78
This work	187	267	219	486	-	564	3.36

Refer to : [1] [8], [2] [9], [3] [1]

NIST Round 3 Comparison Results

Implementation	Timing [cc×10^6]			Memory [B]			Data [B]
	KeyGen	Encaps	Decaps	KeyGen	Encaps	Decaps	pk+ct
Security Level I							
Kyber512	0.46	0.57	0.53	2,396	2,484	2,500	1,568
ntruhps2048509	79.66	0.56	0.54	21,392	14,068	14,800	1,398
lightsaber	0.36	0.49	0.46	5,332	5,292	5,308	1,408
BIKE L1	25.06	3.40	54.79	44,108	32,156	91,400	3,113
FrodoKEM640aes	48.35	47.13	46.59	31,992	62,488	83,104	19,336
FrodoKEM640shake	79.33	79.70	79.15	26,600	51,976	72,592	19,336
SIKEp434compressed	68.26	99.50	74.86	68,636	41,380	7,940	433
	65.4%	47.6%	3.9%				56.1%
SIKEp434	41.28	67.40	72.02	6,108	6,468	6,748	676
Security Level II							
SIKEp503compressed	89.76	143.17	106.26	88,192	53,696	9,008	505
	54.4%	49.9%	4.5%				54.5%
SIKEp503	58.12	95.53	101.73	7,360	7,736	8,112	780
Security Level III							
Kyber768	0.76	0.92	0.86	3,276	2,968	2,988	2,272
ntruhps2048677	143.73	0.82	0.82	28,504	9,036	19,728	1,862
saber	0.66	0.84	0.79	6,364	6,316	6,332	2,080
ntruhrss701	153.10	0.38	0.87	27,560	7,400	20,552	2,276
ntrulpr761	0.74	1.29	1.39	13,168	20,000	24,032	2,206
sntrup761	10.83	0.70	0.57	61,508	13,320	16,952	2,197
SIKEp610compressed	187.67	267.35	219.80	85,740	78,532	11,524	616
	76.9%	37.2%	12.1%				53.9%
SIKEp610	106.07	194.90	196.12	10,490	10,908	11,372	948

For benchmarking results refer to : [1] [4]

Conclusions

- We propose the first integration of compressed SIKE into low-end device with limited resources.
- We reserve additional memory region in the CCM RAM to keep local data structures which overflow the stack.
- We integrate the previous best-reported SIKE results into the compressed SIKE algorithm and obtain 25% and 16% speedup after we implement additional subroutines in assembly language.

References

[1] M. Anastasova, R. Azarderakhsh, and M. M. Kermani. Fast Strategies for the Implementation of SIKE Round 3 on ARM Cortex-M4.

[2] R. Azarderakhsh, D. Jao, K. Kalach, B. Koziel, and C. Leonardi. Key compression for isogeny-based cryptosystems. In *Proceedings of the 3rd ACM International Workshop on ASIA Public-Key Cryptography*, pages 1–10, 2016.

[3] C. Costello, D. Jao, P. Longa, M. Naehrig, J. Renes, and D. Urbanik. Efficient compression of sidh public keys. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques*, pages 679–706. Springer, 2017.

[4] M. J. Kannwischer, J. Rijneveld, P. Schwabe, and K. Stoffelen. pqm4 : Testing and Benchmarking NIST PQC on ARM Cortex-M4. 2019.

[5] M. Naehrig and J. Renes. Dual isogenies and their application to public-key compression for isogeny-based cryptography. In *International Conference on the Theory and Application of Cryptology and Information Security*, pages 243–272. Springer, 2019.

[6] T. N. I. of Standards and T. (NIST). Post-quantum cryptography standardization, 2017-2018. https ://csrc.nist.gov/projects/post-quantum-cryptography/ post-quantum-cryptography-standardization. Last accessed on June 6, 2021.

[7] G. Pereira, J. Doliskani, and D. Jao. x-only point addition formula and faster compressed sike. *Journal of Cryptographic Engineering*, 11(1) :57–69, 2021.

[8] PQCryptov3.3. Sidh library. https ://github.com/Microsoft/PQCrypto-SIDH.

[9] H. Seo, M. Anastasova, A. Jalali, and R. Azarderakhsh. Supersingular Isogeny Key Encapsulation (SIKE) Round 2 on ARM Cortex-M4. *IEEE Transactions on Computers*, 2020.