Argon2 양자회로 구현

https://youtu.be/Kf5CRwGELQk

정보컴퓨터공학과 송경주

Argon2: a memory-hard function for password hashing and other applications

- Argon2은 2015 Password Hashing Competition에서 우승한 key derivation function
- 자격 증명 저장, 키 파생 또는 기타 응용 프로그램을 위한 암호를 해시하는데 사용할 수 있음
- 가장 높은 메모리 채우기 속도와 여러 컴퓨팅 장치의 효과적인 사용을 목표로 하는 단순한 디자인을 가지고 있으면서도 트레이드오프에 대한 기능을 제공함

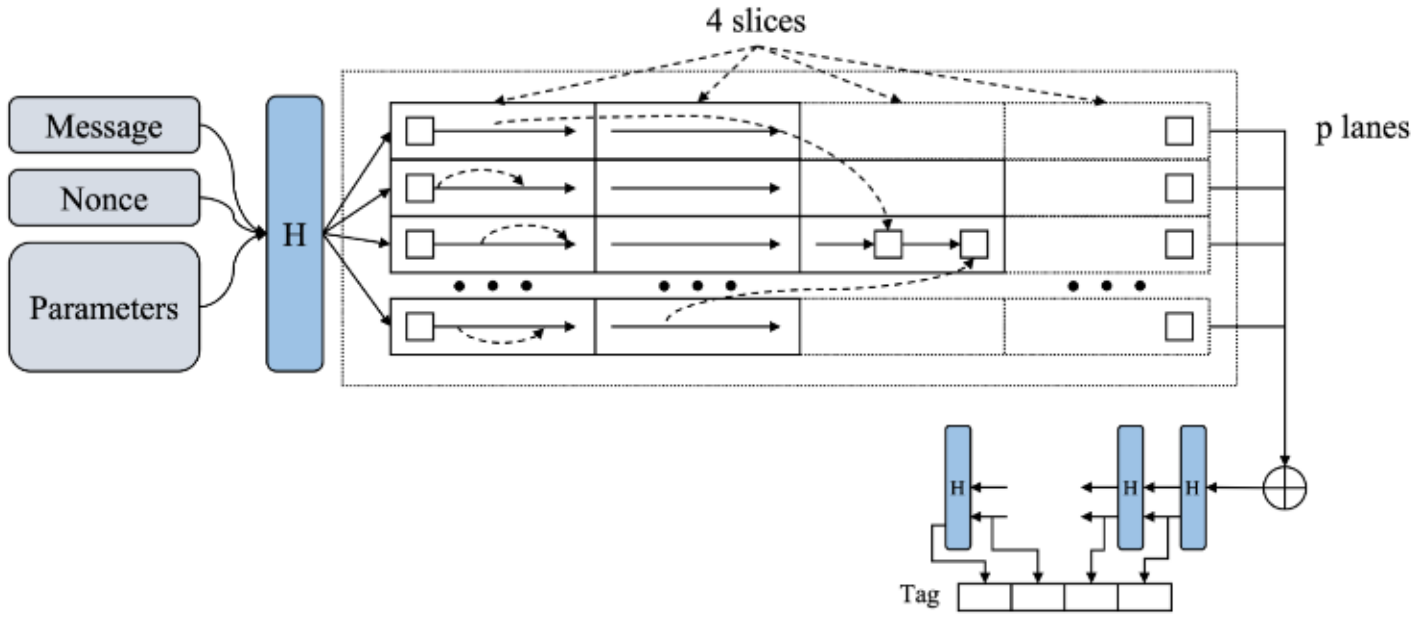
- Argon2는 세가지 변형: Argon2i, Argon2d and Argon2id을 제공함
  1. **Argon2d:** Argon2d는 더 빠르고 데이터에 의존하는 메모리 액세스를 사용하므로 GPU 크래킹 공격에 매우 강하고 사이드 채널 타이밍 공격(예: 암호화폐)의 위협이 없는 애플리케이션에 적합
  2. **Argon2i:** 암호 해싱 및 암호 기반 키 파생에 선호되는 데이터 독립적 메모리 액세스를 사용하지만 트레이드 오프 공격으로부터 보호하기 위해 메모리를 더 많이 통과하므로 속도가 느림
  3. **Argon2id:** Argon2i와 Argon2d의 하이브리드로, 데이터 종속 및 데이터 독립 메모리 액세스의 조합을 사용하여 사이드 채널 캐시 타이밍 공격에 대한 Argon2i의 일부 저항과 GPU 크래킹 공격에 대한 Argon2d의 대부분의 저항을 제공

Argon2: a memory-hard function for password hashing and other applications

- Argon2에는 기본입력과 보조 입력(매개변수)의 두가지 유형의 입력이 있음

  - Primary input:
    Massage P, Nonce S

  - Secondary input:
    Degree of parallelism p
    Tag length τ
    Memory size m
    Number of iterations t
    Version number v
    Secret value K
    Associated data X
    Type y of Argon2



Argon2: a memory-hard function for password hashing and other applications

- Compression function G
  - Argon2에서 사용하는 Compression function G 는 Blake2b 라운드 함수 P를 기반으로 함
  - P는 8개의 16Byte 레지스터(128bit) 입력에서 동작한다.
  - Compression function $G(X, Y)$는 두개의 1024-byte block X, Y로 동작함
  - $R = (X \oplus Y)$연산 후 R을 $R_0 \cdots R_{63}$ 으로 정의됨
  - row-wise 및 column-wise 순서로 P를 적용하여 Z를 얻음

$$
\begin{aligned}
(Q_0, Q_1, \ldots, Q_7) &\leftarrow \mathcal{P}(R_0, R_1, \ldots, R_7); \\
(Q_8, Q_9, \ldots, Q_{15}) &\leftarrow \mathcal{P}(R_8, R_9, \ldots, R_{15}); \\
&\ldots \\
(Q_{56}, Q_{57}, \ldots, Q_{63}) &\leftarrow \mathcal{P}(R_{56}, R_{57}, \ldots, R_{63}); \\
\\
(Z_0, Z_8, Z_{16}, \ldots, Z_{56}) &\leftarrow \mathcal{P}(Q_0, Q_8, Q_{16}, \ldots, Q_{56}); \\
(Z_1, Z_9, Z_{17}, \ldots, Z_{57}) &\leftarrow \mathcal{P}(Q_1, Q_9, Q_{17}, \ldots, Q_{57}); \\
&\ldots \\
(Z_7, Z_{15}, Z_{23}, \ldots, Z_{63}) &\leftarrow \mathcal{P}(Q_7, Q_{15}, Q_{23}, \ldots, Q_{63}).
\end{aligned}
$$

Argon2: a memory-hard function for password hashing and other applications

R_0	R_1	R_2	R_3	R_4	R_5	R_6	R_7
R_8	R_9	R_{10}	R_{11}	R_{12}	R_{13}	R_{14}	R_{15}
R_{16}	R_{17}	R_{18}	R_{19}	R_{20}	R_{21}	R_{22}	R_{23}
R_{24}	R_{25}	R_{26}	R_{27}	R_{28}	R_{29}	R_{30}	R_{31}
R_{32}	R_{33}	R_{34}	R_{35}	R_{36}	R_{37}	R_{38}	R_{39}
R_{40}	R_{41}	R_{42}	R_{43}	R_{44}	R_{45}	R_{46}	R_{47}
R_{48}	R_{49}	R_{50}	R_{51}	R_{52}	R_{53}	R_{54}	R_{55}
R_{56}	R_{57}	R_{58}	R_{59}	R_{60}	R_{61}	R_{62}	R_{63}

P

Q_0	Q_1	Q_2	Q_3	Q_4	Q_5	Q_6	Q_7
Q_8	Q_9	Q_{10}	Q_{11}	Q_{12}	Q_{13}	Q_{14}	Q_{15}
Q_{16}	Q_{17}	Q_{18}	Q_{19}	Q_{20}	Q_{21}	Q_{22}	Q_{23}
Q_{24}	Q_{25}	Q_{26}	Q_{27}	Q_{28}	Q_{29}	Q_{30}	Q_{31}
Q_{32}	Q_{33}	Q_{34}	Q_{35}	Q_{36}	Q_{37}	Q_{38}	Q_{39}
Q_{40}	Q_{41}	Q_{42}	Q_{43}	Q_{44}	Q_{45}	Q_{46}	Q_{47}
Q_{48}	Q_{49}	Q_{50}	Q_{51}	Q_{52}	Q_{53}	Q_{54}	Q_{55}
Q_{56}	Q_{57}	Q_{58}	Q_{59}	Q_{60}	Q_{61}	Q_{62}	Q_{63}

Argon2: a memory-hard function for password hashing and other applications

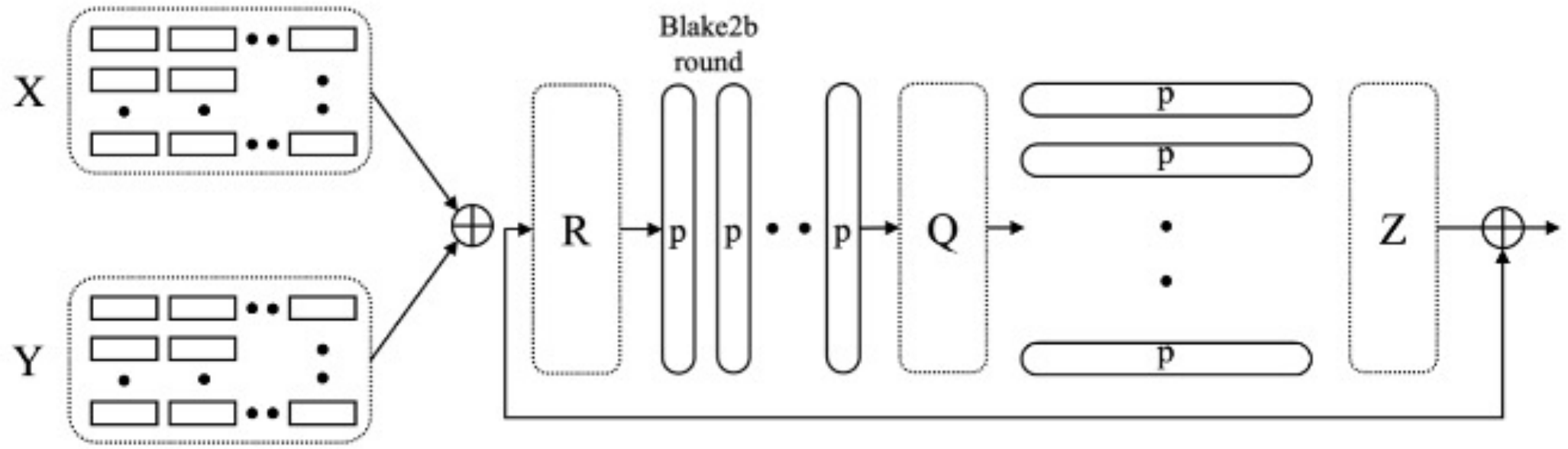
Q_0	Q_1	Q_2	Q_3	Q_4	Q_5	Q_6	Q_7
Q_8	Q_9	Q_{10}	Q_{11}	Q_{12}	Q_{13}	Q_{14}	Q_{15}
Q_{16}	Q_{17}	Q_{18}	Q_{19}	Q_{20}	Q_{21}	Q_{22}	Q_{23}
Q_{24}	Q_{25}	Q_{26}	Q_{27}	Q_{28}	Q_{29}	Q_{30}	Q_{31}
Q_{32}	Q_{33}	Q_{34}	Q_{35}	Q_{36}	Q_{37}	Q_{38}	Q_{39}
Q_{40}	Q_{41}	Q_{42}	Q_{43}	Q_{44}	Q_{45}	Q_{46}	Q_{47}
Q_{48}	Q_{49}	Q_{50}	Q_{51}	Q_{52}	Q_{53}	Q_{54}	Q_{55}
Q_{56}	Q_{57}	Q_{58}	Q_{59}	Q_{60}	Q_{61}	Q_{62}	Q_{63}

P

Z_0	Z_1	Z_2	Z_3	Z_4	Z_5	Z_6	Z_7
Z_8	Z_9	Z_{10}	Z_{11}	Z_{12}	Z_{13}	Z_{14}	Z_{15}
Z_{16}	Z_{17}	Z_{18}	Z_{19}	Z_{20}	Z_{21}	Z_{22}	Z_{23}
Z_{24}	Z_{25}	Z_{26}	Z_{27}	Z_{28}	Z_{29}	Z_{30}	Z_{31}
Z_{32}	Z_{33}	Z_{34}	Z_{35}	Z_{36}	Z_{37}	Z_{38}	Z_{39}
Z_{40}	Z_{41}	Z_{42}	Z_{43}	Z_{44}	Z_{45}	Z_{46}	Z_{47}
Z_{48}	Z_{49}	Z_{50}	Z_{51}	Z_{52}	Z_{53}	Z_{54}	Z_{55}
Z_{56}	Z_{57}	Z_{58}	Z_{59}	Z_{60}	Z_{61}	Z_{62}	Z_{63}

Argon2: a memory-hard function for password hashing and other applications

- Compression function G



Argon2: a memory-hard function for password hashing and other applications

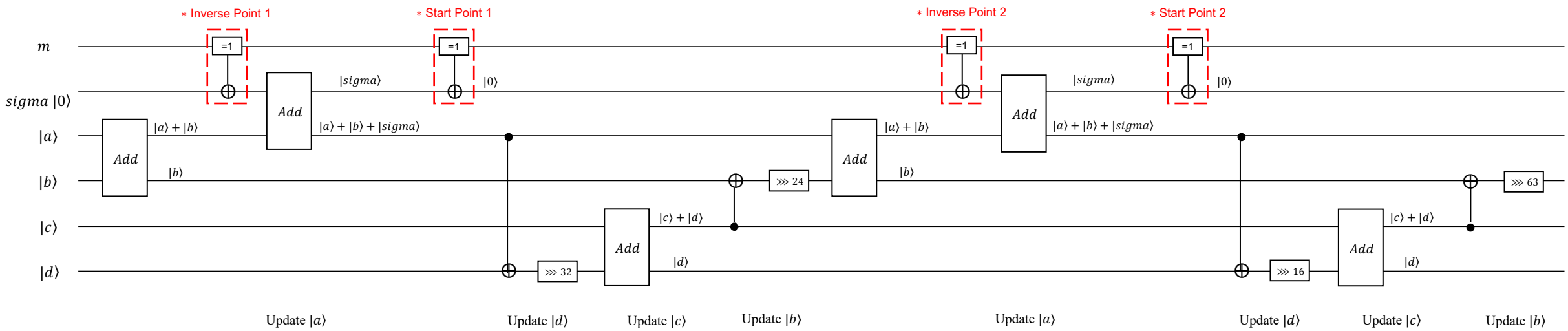
- Permutation P
- 8개의 16-byte input $S_0, S_1 \cdots S_7$가 64bit 워드의 $4\times4\ matrix$로 표현된다.
  - $S_i = (v_{2i+1}||v_{2i})$

$$\begin{pmatrix} v_0 & v_1 & v_2 & v_3 \\ v_4 & v_5 & v_6 & v_7 \\ v_8 & v_9 & v_{10} & v_{11} \\ v_{12} & v_{13} & v_{14} & v_{15} \end{pmatrix}$$

$$\begin{aligned} G(v_0, v_4, v_8, v_{12}) \quad & G(v_1, v_5, v_9, v_{13}) \\ G(v_2, v_6, v_{10}, v_{14}) \quad & G(v_3, v_7, v_{11}, v_{15}) \\ G(v_0, v_5, v_{10}, v_{15}) \quad & G(v_1, v_6, v_{11}, v_{12}) \\ G(v_2, v_7, v_8, v_{13}) \quad & G(v_3, v_4, v_9, v_{14}), \end{aligned}$$

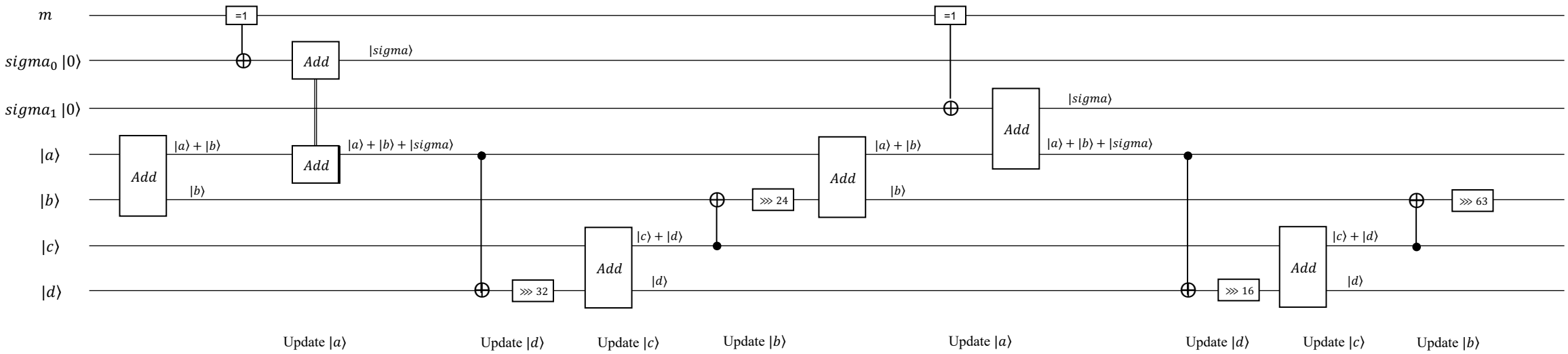
$$\begin{aligned} a &\leftarrow a + b + 2 * a_L * b_L; \\ d &\leftarrow (d \oplus a) \ggg 32; \\ c &\leftarrow c + d + 2 * c_L * d_L; \\ b &\leftarrow (b \oplus c) \ggg 24; \\ a &\leftarrow a + b + 2 * a_L * b_L; \\ d &\leftarrow (d \oplus a) \ggg 16; \\ c &\leftarrow c + d + 2 * c_L * d_L; \\ b &\leftarrow (b \oplus c) \ggg 63; \end{aligned}$$

Argon2 양자회로 구현



(1) Qubit-optimized Quantum circuit for G

Argon2 양자회로 구현



(2) Depth-optimized Quantum circuit for G

Argon2 양자회로 구현

Operation	Adder	#Qubit	#1qClifford	#CNOT	#Toffoli	#Full Depth
G (Qubit Opt)	Ripple	1,089	70,836	204,864	72,000	74,713
G (Qubit Opt)	Simple	1,089	22	172,032	72,576	220,033
G (Depth Opt)	Ripple	13,318	70,284	204,864	72,000	23,401
G (Depth Opt)	Simple	13,318	12	172,032	72,576	68,163
$Z \oplus R$	-	1,536	-	1,024	-	2

Function	#Qubit	#1qClifford	#CNOT	#Toffoli	#Full Depth
Initial	1,090	(Not used.)			
Update		(Not used.)			
Final		1.62×2^{17}	1.17×2^{19}	1.64×2^{17}	1.71×2^{17}
blake2b		1.51×2^{22}	1.1×2^{24}	1.54×2^{22}	1.6×2^{22}
Total		1.51×2^{22}	1.14×2^{24}	1.59×2^{22}	1.65×2^{22}

Q & A