Cryptographic Schemes under Quantum Supremacy and IoT

National Taipei University

Assistant Professor

Shiu, Hung-Jr (許宏誌)

Outline

- Threats
- Solutions

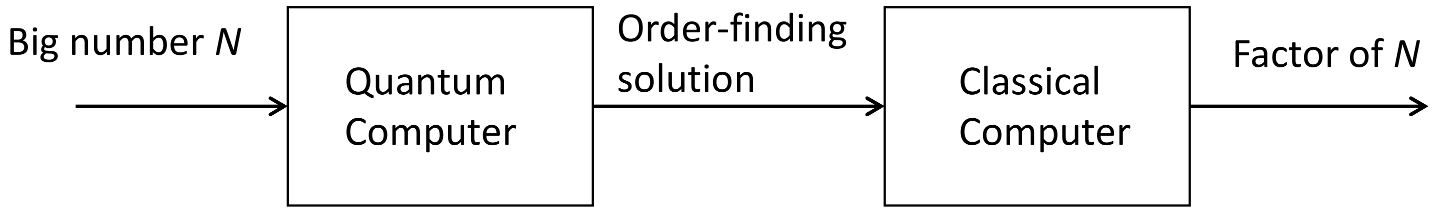
Threats

- Quantum Supremacy
  - Shor's Algorithm
    - Threats to public-key crypto-systems
  - Grover's Algorithm
    - Security-level shrinkage of private-key crypto-systems

- Internet Of Things
  - Side-Channel Attack for private-key crypto-systems

Shor's Algorithm

- Input: an odd composite number N with n bits
- Output: a non-trivial factorization of N with some probability
- quantum parallelism
- Reduce to order-finding problem



Complexity

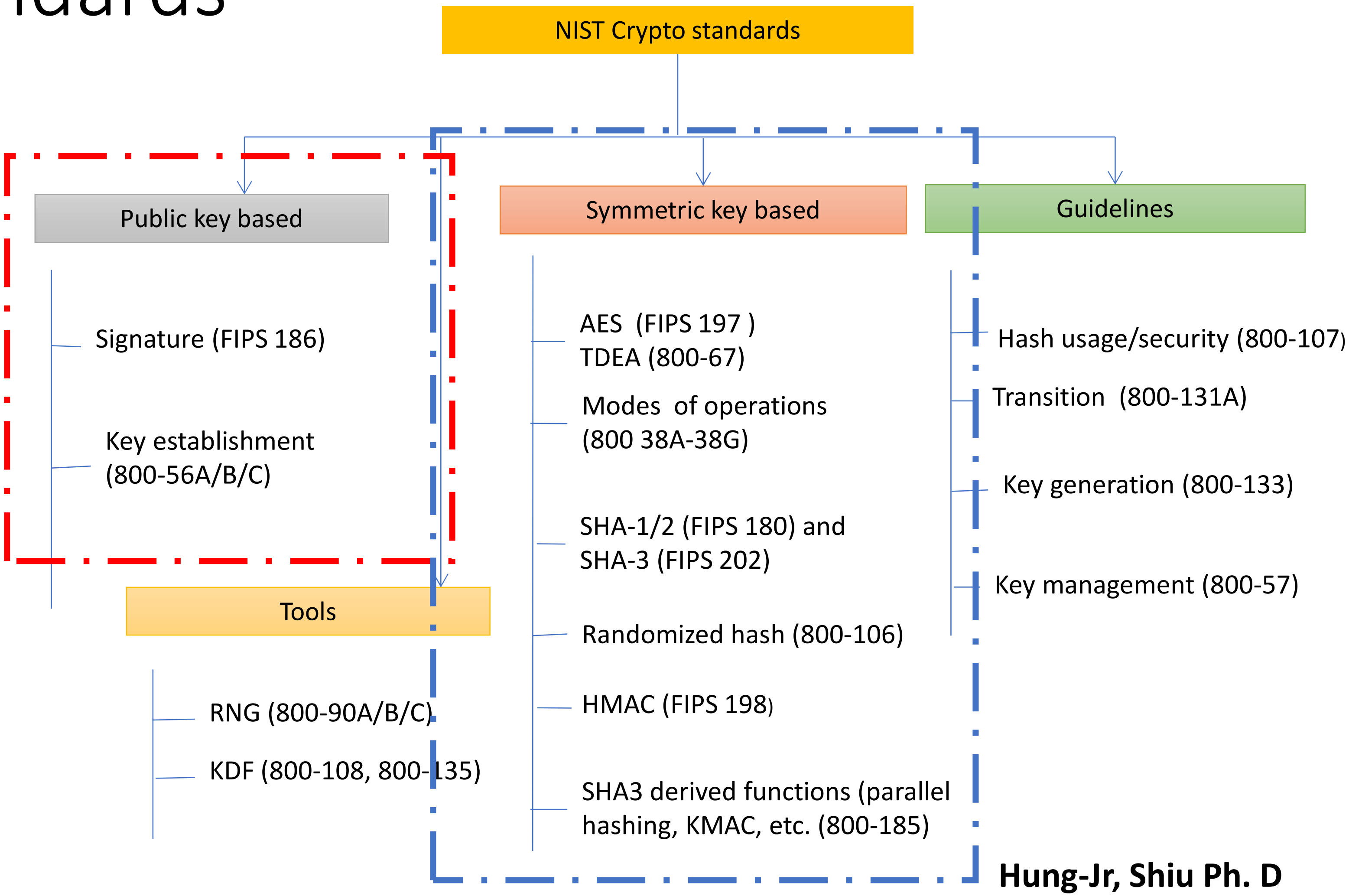
- Modular exponential: $\boldsymbol{\Theta}(n^3)$
- Quantum Fourier Transform (QFT):$\boldsymbol{\Theta}(n^2)$
- Succeed probability: $\boldsymbol{\Omega}(1/\log n)$
- Total time complexity is $\mathrm{O}(n^3 \log n)$.
- Need $3n$ qubits in total.

Q. Threats

algorithm\security	2^{80}	2^{112}	2^{128}	2^{192}	2^{256}
RSA key length (bits)	1024	2048	3072	7680	15360
ECC key length (bits)	160	224	256	384	512
Key length ratio	6:1	9:1	12:1	20:1	30:1
RSA signature length (bytes)	X	X	384	960	1920
ECDSA length (bytes)	X	X	64	96	132
qubits to break (RSA)	3072	6144	9216	23040	46080
qubits to break (ECC)	480	672	768	1152	1536

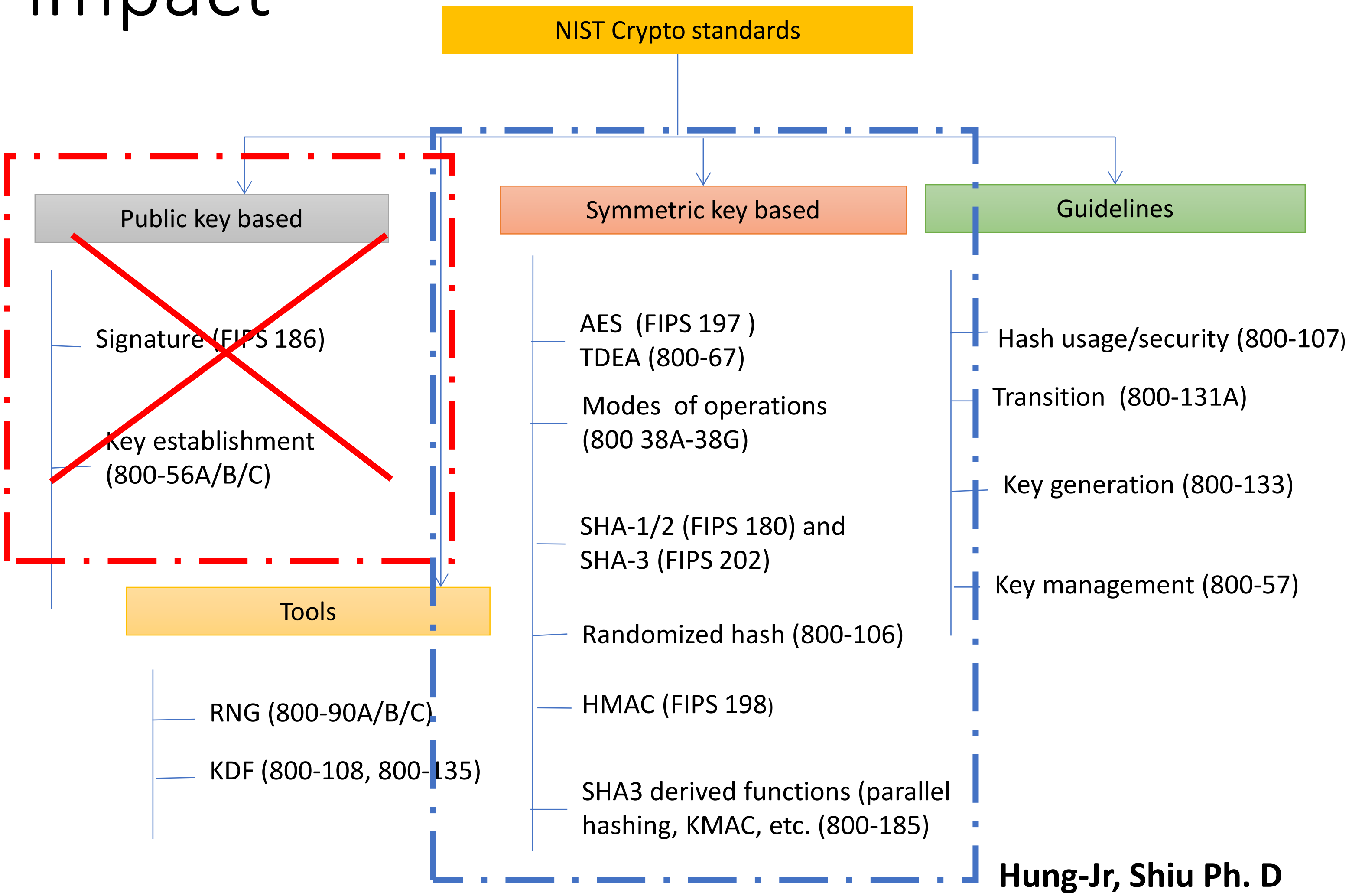
Google and IBM now have 50~70 qubits

Standards



Hung-Jr, Shiu Ph. D

The impact



Hung-Jr, Shiu Ph. D

Grover's Algorithm

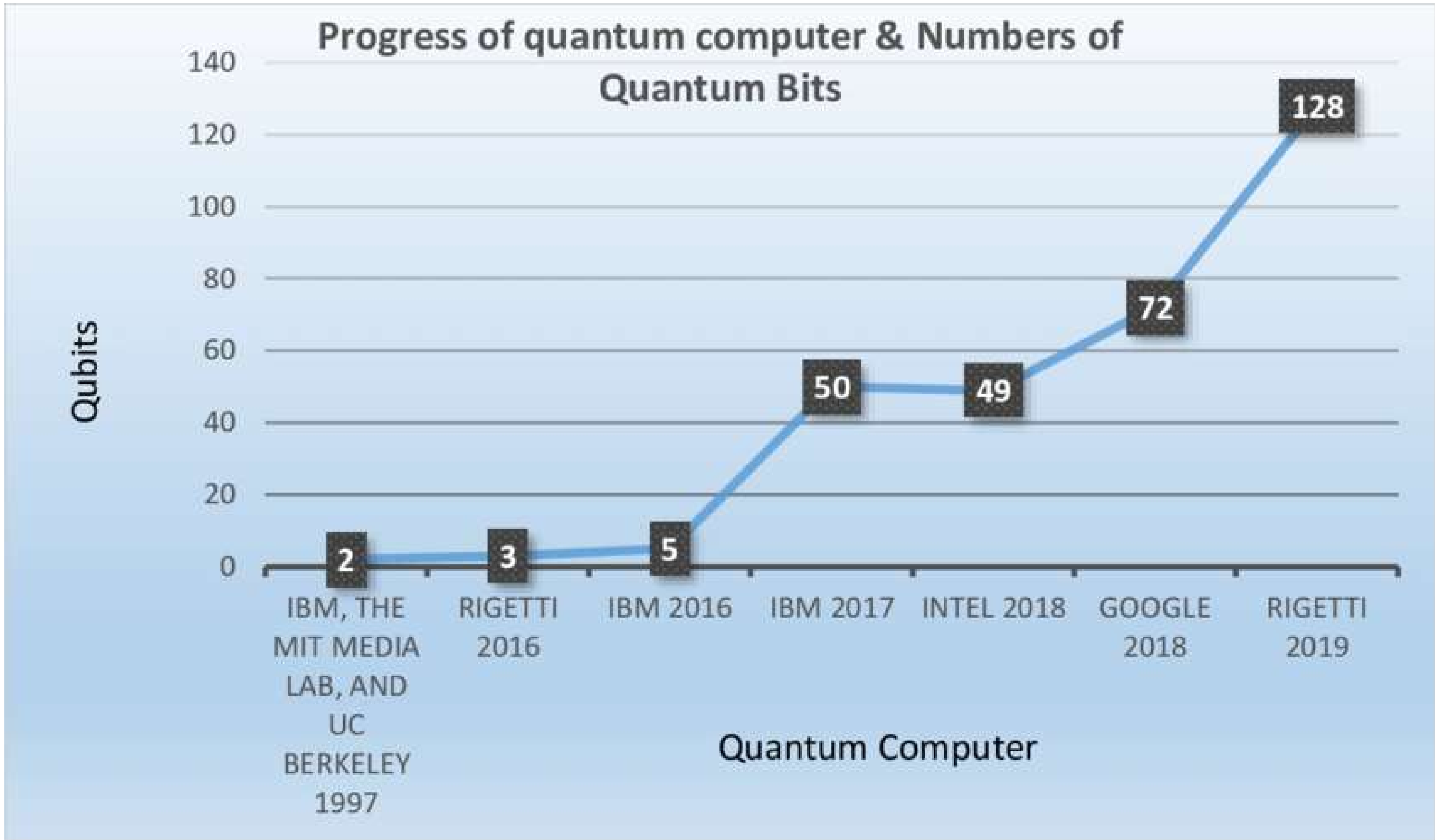
- Search database with probabilities.
- $O(N) \rightarrow O(\sqrt{N})$
- The security level of AES128 will shrink from 2^{128} to 2^{64}, AES256 from 2^{256} to 2^{128}.

Security Level

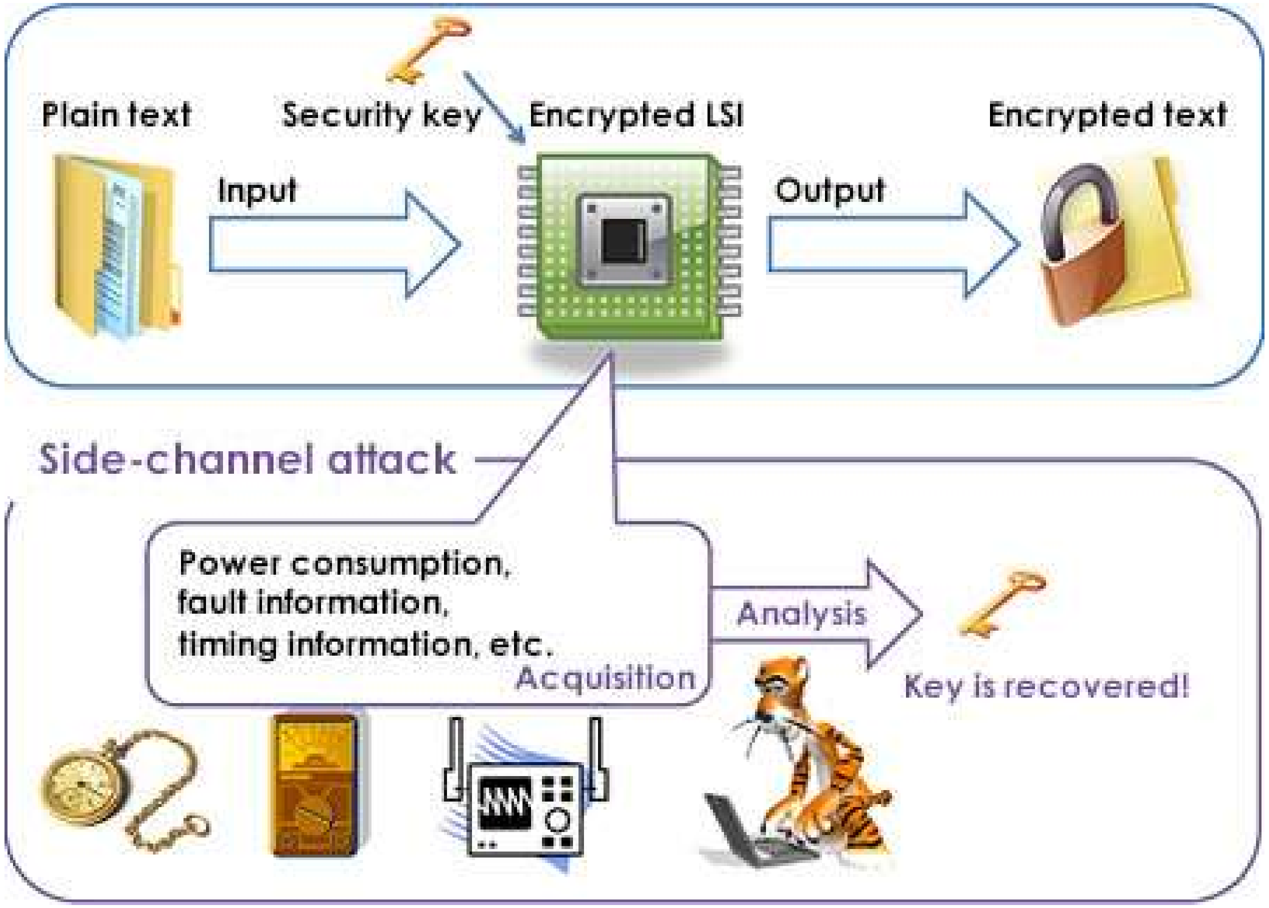
Level	Security Description
I	At least as hard to break as AES128 (exhaustive key search)
II	At least as hard to break as SHA256 (collision search)
III	At least as hard to break as AES192 (exhaustive key search)
IV	At least as hard to break as SHA384 (collision search)
V	At least as hard to break as AES256 (exhaustive key search)

- Only level I still survive under Grover's algorithm.

Qbits trend.

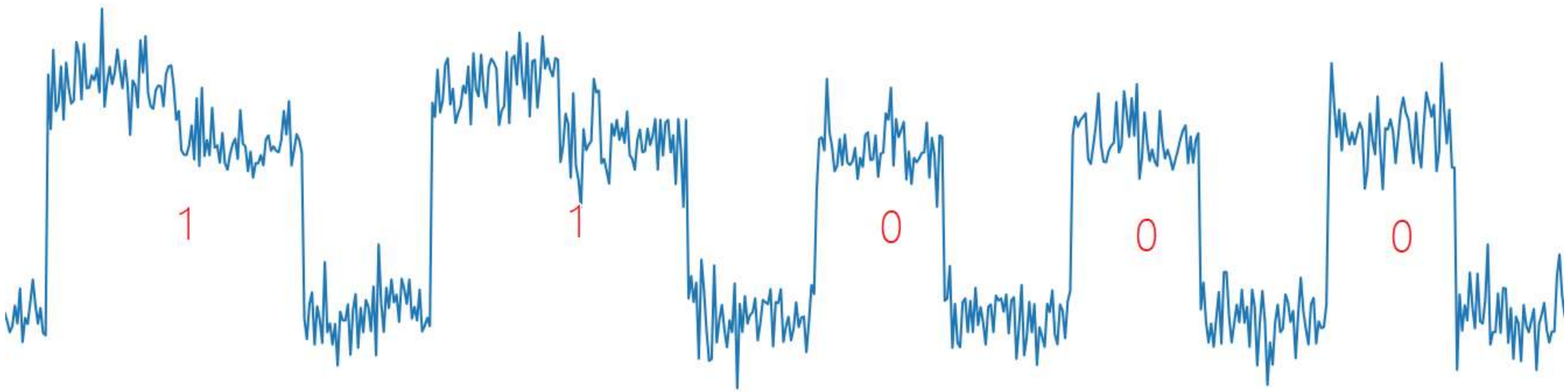

Progress of quantum computer & Numbers of Quantum Bits
Qubits
140
120
100
80
60
40
20
0
2
3
5
50
49
72
128
IBM, THE MIT MEDIA LAB, AND UC BERKELEY 1997
RIGETTI 2016
IBM 2016
IBM 2017
INTEL 2018
GOOGLE 2018
RIGETTI 2019
Quantum Computer

Side-channel attacks



A RSA SCA example

- Observing the multiplication



Solutions

- For quantum computing:
  - public-key crypto-systems:
    - Post Quantum Cryptography
  - private-key crypto-systems:
    - AES 384/512 (extending the key length)
- For side-channel attack in IoT:
  - Customized AES
  - Signal masking
  - Pipeline

PQC Timeline

- Aug 2016 – Draft submission requirements & evaluation criteria
- Dec 2016 – Final requirements and criteria
- Nov 2017 – Deadline for submissions
- Apr 2018 – NIST PQC Workshop – submitters' presentations
- 2018/2019 – 2nd Round begins (smaller number of submissions)
  - minor changes allowed
- Aug 2019 – 2nd NIST PQC Workshop
- 2020/2021 - Select algorithms or start a 3rd Round
- 2022-2024 - Draft standards available

- NIST will release reports on progress and selection rationale

Submissions

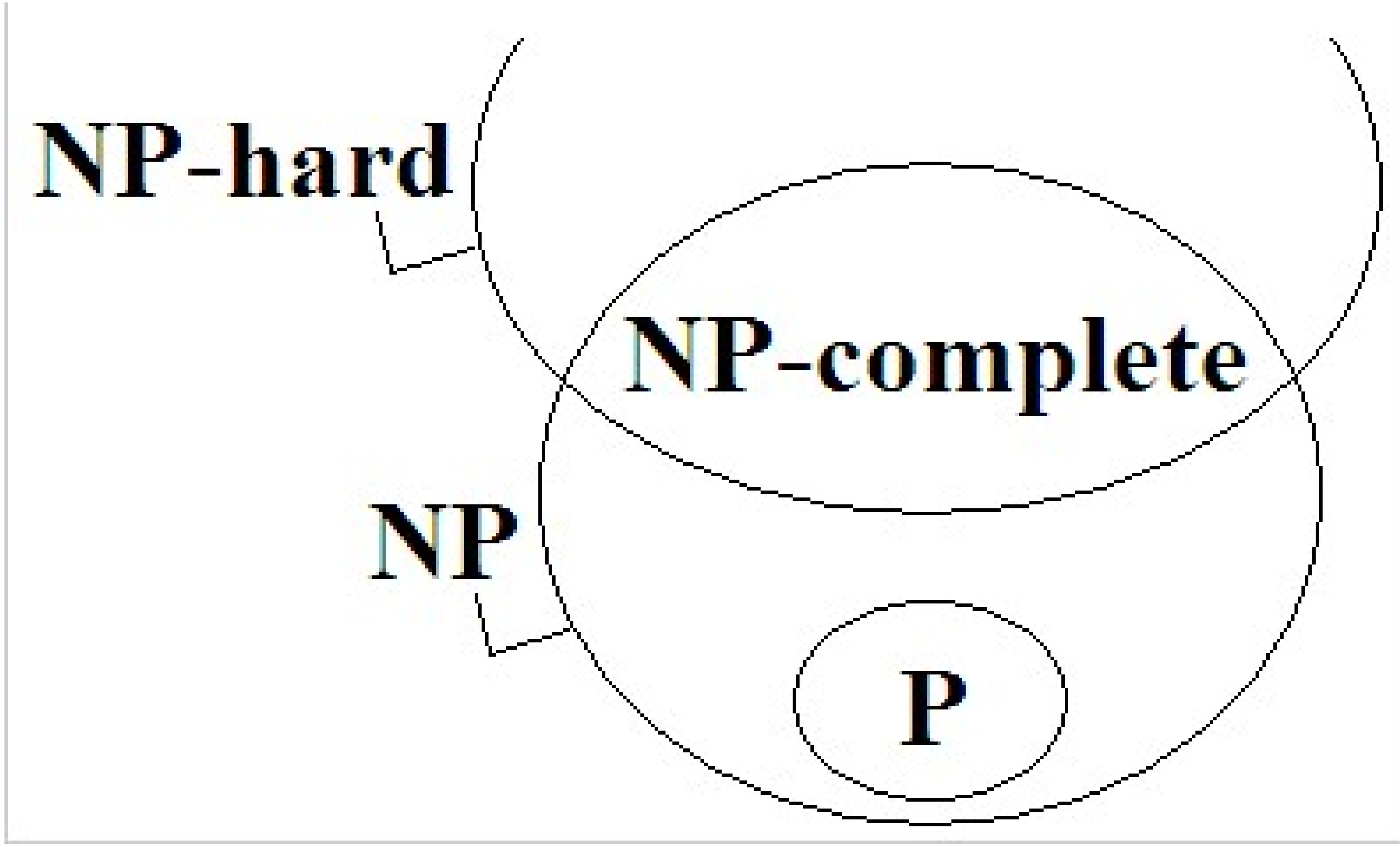
	Signatures	KEM/Encryption	Overall
Lattice-based	5	21	26
Code-based	2	17	19
Multi-variate	7	2	9
Symmetric/Hash-based	3		3
Other	2	5	7
Total	**19**	**45**	**64**

Round 3

- Candidate: (7)
- Key establishment
  - Classic McEllece
  - CRYSTALS-KYBER
  - NTRU
  - SABER
- Digital signature
  - CRYSTALS-DILITHIUM
  - FALCON
  - Rainbow

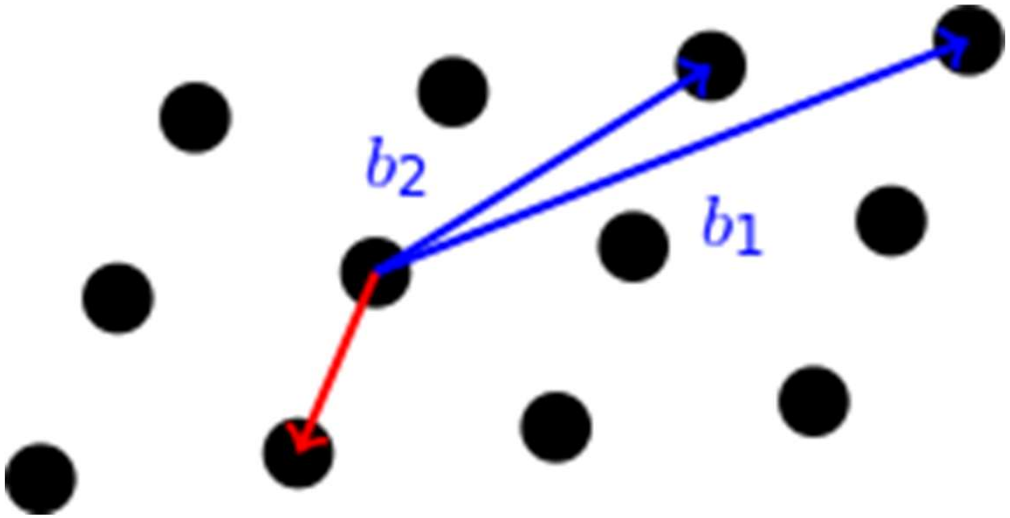
- Alternative candidate: (8)
- Key establishment
  - BIKE
  - FrodoKEM
  - HQC
  - NTRU Prime
  - SIKE
- Digital signature
  - GeMSS
  - Picnic
  - SPHINCS+

Hardness of problems



Lattice

- Shortest Vector Problem (NP hard)
  - Given basis of a vector space *V* and a lattice *L* with a norm *N*, find the shortest non-zero vector in *V*.



Recall RSA: Key Generation

- Alice:
  - choose $N = p \times q$, p, q are prime and $p \neq q$
  - find $r = (p - 1) \times (q - 1)$
  - choose e, $e < r$ and find d, $e \times d \equiv_r 1$
  - release (N, e) as the public key
  - obtain (N, d) as the private key

Recall RSA: Encryption/Decryption

- Encryption: (Bob)
  - plaintext $n < N$
  - cipher $c \equiv_N n^e$
  - deliver c
- Decryption: (Alice)
  - $n \equiv_N c^d \equiv_N n^{ed} (ed = 1)$

NTRUEncryption (1/9)

- **<u>Definition</u>**. For any positive integers d_1, d_2, define

$$\left\{ \boldsymbol{a}(x) \in R : \begin{array}{l} \boldsymbol{a}(x) \text{ has } d_1 \text{ coefficients equal to } 1, \\ \boldsymbol{a}(x) \text{ has } d_2 \text{ coefficients equal to } -1, \\ \boldsymbol{a}(x) \text{ has all other coefficients equal to } 0 \end{array} \right\}$$

- Polynomials in $\mathcal{T}(d_1, d_2)$ are called *ternary* (or *trinary*) *polynomials*

NTRUEncryption (2/9)

- Public parameter creation
  - Alice (or some trusted authority) chooses public parameters (N, p, q, d) satisfying
    - N, p: primes
    - $\gcd(N, q) = \gcd(p, q) = 1$
    - $q > (6d + 1)p$
  - The third requirement assures the correctness of the decryption

NTRUEncryption (3/9)

- Key creation
  - Alice's private key consists of two randomly chosen polynomials
    - $\boldsymbol{f}(x) \in \mathcal{T}(d+1, d)$
    - $\boldsymbol{g}(x) \in \mathcal{T}(d, d)$
  - Also Alice computes the inverses
    - $\boldsymbol{F}_q(x) = \boldsymbol{f}(x)^{-1} \in R_q$
    - $\boldsymbol{F}_p(x) = \boldsymbol{f}(x)^{-1} \in R_p$

NTRUEncryption (4/9)

- If either inverse fails to exist, choose a new $\boldsymbol{f}(x)$
- Note that elements in $\mathcal{T}(d, d)$ never have inverses in R_q
- Alice next computes the public key $\boldsymbol{h}(x) \in R_q$ by

$$\boldsymbol{h}(x) = \boldsymbol{F}_q(x) \star \boldsymbol{g}(x)$$

- To decrypt messages, Alice needs $\boldsymbol{f}(x)$ and $\boldsymbol{F}_p(x)$ She can store only $\boldsymbol{f}(x)$ and re-compute $\boldsymbol{F}_p(x)$ when she needs it

NTRUEncryption (5/9)

- Encryption
  - Bob's plaintext is a polynomial $\boldsymbol{m}(x) \in R$ whose coefficients satisfy $-\frac{1}{2}p < m_i \leq \frac{1}{2}p$
    - The plaintext $\boldsymbol{m}$ is a polynomial in R that is the center-lift of a polynomial in R_p
  - Bob chooses a random polynomial (a random element) $\boldsymbol{r}(x) \in \mathcal{T}(d, d)$
  - Bob computes ciphertext $\boldsymbol{e}(x) \in R_q$ by

$$\boldsymbol{e}(x) \equiv p\boldsymbol{h}(x) \star \boldsymbol{r}(x) + \boldsymbol{m}(x) \pmod{q}$$

NTRUEncryption (6/9)

- Decryption
  - On receiving Bob's ciphertext, Alice starts the decryption process by computing
    $$\boldsymbol{a}(x) \equiv \boldsymbol{f}(x) \star \boldsymbol{e}(x) \pmod{q}$$
  - Then center-lifts $\boldsymbol{a}(x)$ to an element of R and does a $mod\ p$ computation
    $$\boldsymbol{b}(x) \equiv \boldsymbol{F}_p(x) \star \boldsymbol{a}(x) \pmod{p}$$
  - The polynomial $\boldsymbol{b}(x)$ is equal to the plaintext $\boldsymbol{m}(x)$

NTRUEncryption (7/9)

Public parameter creation	
A trusted party chooses public parameters (N, p, q, d) with N and p prime, $\gcd(p, q) = \gcd(N, q) = 1$, and $q > (6d+1)p$.	
Alice	**Bob**
Key creation	
Choose private $\boldsymbol{f} \in \mathcal{T}(d+1, d)$ that is invertible in R_q and R_p. Choose private $\boldsymbol{g} \in \mathcal{T}(d, d)$. Compute $\boldsymbol{F}_q$, the inverse of $\boldsymbol{f}$ in R_q. Compute $\boldsymbol{F}_p$, the inverse of $\boldsymbol{f}$ in R_p. Publish the public key $\boldsymbol{h} = \boldsymbol{F}_q \star \boldsymbol{g}$.	
Encryption	
	Choose plaintext $\boldsymbol{m} \in R_p$. Choose a random $\boldsymbol{r} \in \mathcal{T}(d, d)$. Use Alice's public key $\boldsymbol{h}$ to compute $\boldsymbol{e} \equiv p\boldsymbol{r} \star \boldsymbol{h} + \boldsymbol{m} \pmod q$. Send ciphertext $\boldsymbol{e}$ to Alice.
Decryption	
Compute $\boldsymbol{f} \star \boldsymbol{e} \equiv p\boldsymbol{g} \star \boldsymbol{r} + \boldsymbol{f} \star \boldsymbol{m} \pmod q$. Center-lift to $\boldsymbol{a} \in R$ and compute $\boldsymbol{m} \equiv \boldsymbol{F}_p \star \boldsymbol{a} \pmod p$.	

NTRUEncryption (8/9)

- Even if the largest coefficient of $g(x)*r(x)$ happens to coincide with the largest coefficient of $f(x) * m(x)$, the largest coefficient of has magnitude at most

$$p\boldsymbol{g}(x) \star \boldsymbol{r}(x) + \boldsymbol{f}(x) \star \boldsymbol{m}(x)$$

$$p \cdot 2d + (2d+1) \cdot \frac{1}{2}p = \left(3d + \frac{1}{2}\right)p$$

NTRUEncryption (9/9)

- When Alice computes $\boldsymbol{a}(x)$ modulo q and then lifts it to R, she recovers the exact value
- Finally,

$$
\begin{aligned}
&\equiv \boldsymbol{F}_p(x) \star \left(p\boldsymbol{g}(x) \star \boldsymbol{r}(x) + \boldsymbol{f}(x) \star \boldsymbol{m}(x)\right) \pmod{p} \\
&\equiv \boldsymbol{F}_p(x) \star \boldsymbol{f}(x) \star \boldsymbol{m}(x) \pmod{p} \\
&\equiv \boldsymbol{m}(x) \pmod{p}.
\end{aligned}
$$

Lattice and NTRUEncryption

- Finding $g(x)$ and $f(x)$ from $h(x)$ is hard

- $h(x)$ can be seen as given basis, $g(x)$ and $f(x)$ can be seen as shortest vectors.

Code based

- Coding problem is NP hard:

- Input: A binary matrix H, a binary vector s and a nonnegative integer w.
- Question: Is there a vector x of Hamming weight <= w such that $Hx=s$?

McEliece (1/13)

- Given a generator matrix ***G*** & parity-check matrix ***H***
  - e.g. (7,4) Hamming code, ***G*** is a $k \times n$ matrix

$$G = \begin{pmatrix} 1 & 0 & 0 & 0 & 1 & 1 & 0 \\ 0 & 1 & 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 \end{pmatrix}$$

$$H = \begin{pmatrix} 1 & 1 & 0 & 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 1 & 0 & 1 & 0 \\ 0 & 1 & 1 & 1 & 0 & 0 & 1 \end{pmatrix}$$

McEliece (2/13)

- Bob chooses a scrambler matrix ***S***
- ***S*** is a $k \times k$ invertible matrix

$$S = \begin{pmatrix} 1 & 1 & 0 & 1 \\ 1 & 0 & 0 & 1 \\ 0 & 1 & 1 & 1 \\ 1 & 1 & 0 & 0 \end{pmatrix} \qquad S^{-1} = \begin{pmatrix} 1 & 1 & 0 & 1 \\ 1 & 1 & 0 & 0 \\ 0 & 1 & 1 & 1 \\ 1 & 0 & 0 & 1 \end{pmatrix}$$

McEliece (3/13)

- Bob chooses a permutation matrix ***P***
- ***P*** is an $n \times n$ permutation matrix

$$P = \begin{pmatrix} 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 \end{pmatrix}$$

McEliece (4/13)

- Bob publish public key (***G'*** , t) , ***G'*** = ***SGP***
- Bob has the private key (***S***, D_g, ***P***)

$$G' = SGP = \begin{pmatrix} 1 & 1 & 0 & 1 \\ 1 & 0 & 0 & 1 \\ 0 & 1 & 1 & 1 \\ 1 & 1 & 0 & 0 \end{pmatrix} \begin{pmatrix} 1 & 0 & 0 & 0 & 1 & 1 & 0 \\ 0 & 1 & 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 \end{pmatrix} \begin{pmatrix} 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 \end{pmatrix}$$

McEliece (5/13)

- Bob publish public key (***G'*** , t) , ***G'*** = ***SGP***
  - t is the error correcting ability
- Bob has the private key (***S***, D_g, ***P***)
  - D_g is a fast decoding Algorithm

$$G' = \begin{pmatrix} 1 & 1 & 1 & 1 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 1 & 1 & 0 & 1 \\ 0 & 1 & 0 & 1 & 1 & 1 & 0 \end{pmatrix}$$

McEliece (6/13)

- If Alice wishes to send a message ***x*** = (1,1,0,1)
- Alice randomly choose an error vector with weight 1
- e.g. ***e*** = (0, 0, 0, 0, 1, 0, 0)
- Compute ***c*** = ***x G'*** + ***e***

$$c = \begin{pmatrix} 0 & 1 & 1 & 0 & 1 & 1 & 0 \end{pmatrix}$$

$$xG'+e = \begin{pmatrix} 1 & 1 & 0 & 1 \end{pmatrix} \begin{pmatrix} 1 & 1 & 1 & 1 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 1 & 1 & 0 & 1 \\ 0 & 1 & 0 & 1 & 1 & 1 & 0 \end{pmatrix} + \begin{pmatrix} 0 & 0 & 0 & 0 & 1 & 0 & 0 \end{pmatrix}$$

McEliece (7/13)

- Bob receive $\boldsymbol{c} = \boldsymbol{xG'} + \boldsymbol{e}$
- First, compute $\boldsymbol{c'} = \boldsymbol{cP^{-1}} = \boldsymbol{xSG} + \boldsymbol{eP^{-1}}$

$$c' = \begin{pmatrix} 1 & 0 & 0 & 0 & 1 & 1 & 1 \end{pmatrix}$$

$$cP^{-1} = \begin{pmatrix} 0 & 1 & 1 & 0 & 1 & 1 & 0 \end{pmatrix} \begin{pmatrix} 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 \end{pmatrix}$$

McEliece (8/13)

- Bob apply D_g, to find and remove error from $\boldsymbol{c'}$
- $\boldsymbol{c'H^T = (xSG+eP^{-1})H^T = xS\cancel{GH^T} + eP^{-1}H^T = eP^{-1}H^T}$

$$GH^T = \begin{pmatrix} 1 & 0 & 0 & 0 & 1 & 1 & 0 \\ 0 & 1 & 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 \end{pmatrix} \begin{pmatrix} 1 & 1 & 0 \\ 1 & 0 & 1 \\ 0 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix} = 0$$

McEliece (9/13)

- Bob find and remove error from ***c'***
- ***c'H^T = (xSG+eP^-1)H^T = xSGH^T + eP^-1H^T = eP^-1H^T***

$$c'H^T = \begin{pmatrix}1 & 0 & 0 & 0 & 1 & 1 & 1\end{pmatrix}\begin{pmatrix}1 & 1 & 0\\ 1 & 0 & 1\\ 0 & 1 & 1\\ 1 & 1 & 1\\ 1 & 0 & 0\\ 0 & 1 & 0\\ \boxed{0 \quad 0 \quad 1}\end{pmatrix} = \boxed{\begin{pmatrix}0 & 0 & 1\end{pmatrix}}$$

McEliece (10/13)

- Bob find and remove error from ***c'***
- ***c'H^T^ = (xSG+eP^-1^)H^T^ = xSGH^T^ + eP^-1^H^T^ = eP^-1^H^T^***

$$\begin{pmatrix} 0 & 0 & 1 \end{pmatrix} = \left(eP^{-1}\right)H^{T} = \boxed{\begin{pmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix}} \begin{pmatrix} 1 & 1 & 0 \\ 1 & 0 & 1 \\ 0 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

McEliece (11/13)

- Bob find and remove error from ***c'***
- ***xSG = c' - eP⁻¹ = (xSG + eP⁻¹) - eP⁻¹***

$$xSG = c' - eP^{-1}$$

$$= (1 \quad 0 \quad 0 \quad 0 \quad 1 \quad 1 \quad 1) - (0 \quad 0 \quad 0 \quad 0 \quad 0 \quad 0 \quad 1)$$

$$= (1 \quad 0 \quad 0 \quad 0 \quad 1 \quad 1 \quad 0)$$

McEliece (12/13)

- Bob recover ***xS*** = (1, 0, 0, 0)
- Recall the structure of ***G***

$$G = \begin{pmatrix} 1 & 0 & 0 & 0 & 1 & 1 & 0 \\ 0 & 1 & 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 \end{pmatrix}$$

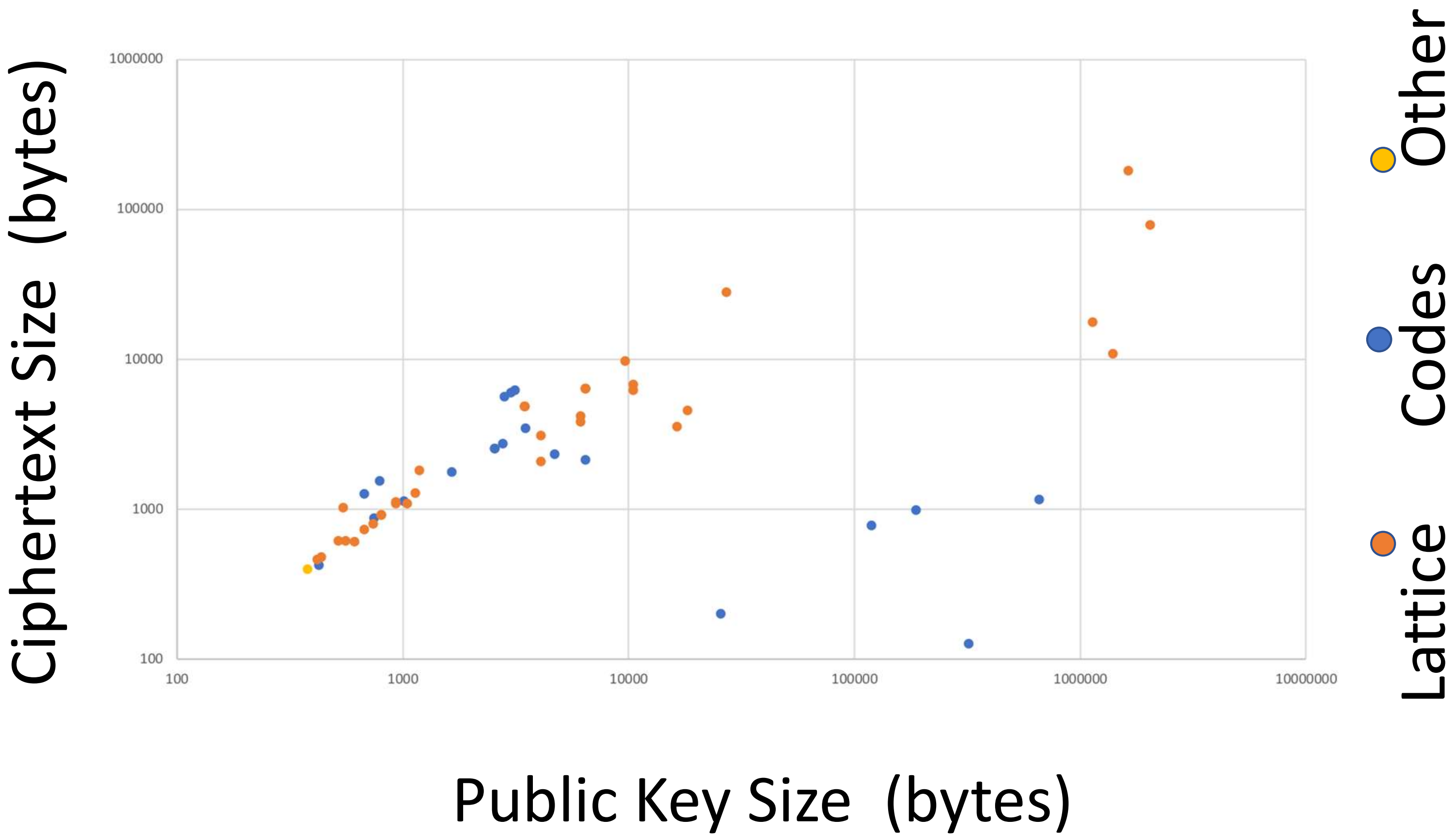
$$xSG = \begin{pmatrix} \boxed{1 \quad 0 \quad 0 \quad 0} & 1 & 1 & 0 \end{pmatrix}$$

McEliece (13/13)

- Bob decode ***x = (xS)S***$^{-1}$ **=** (1, 1, 0, 1)

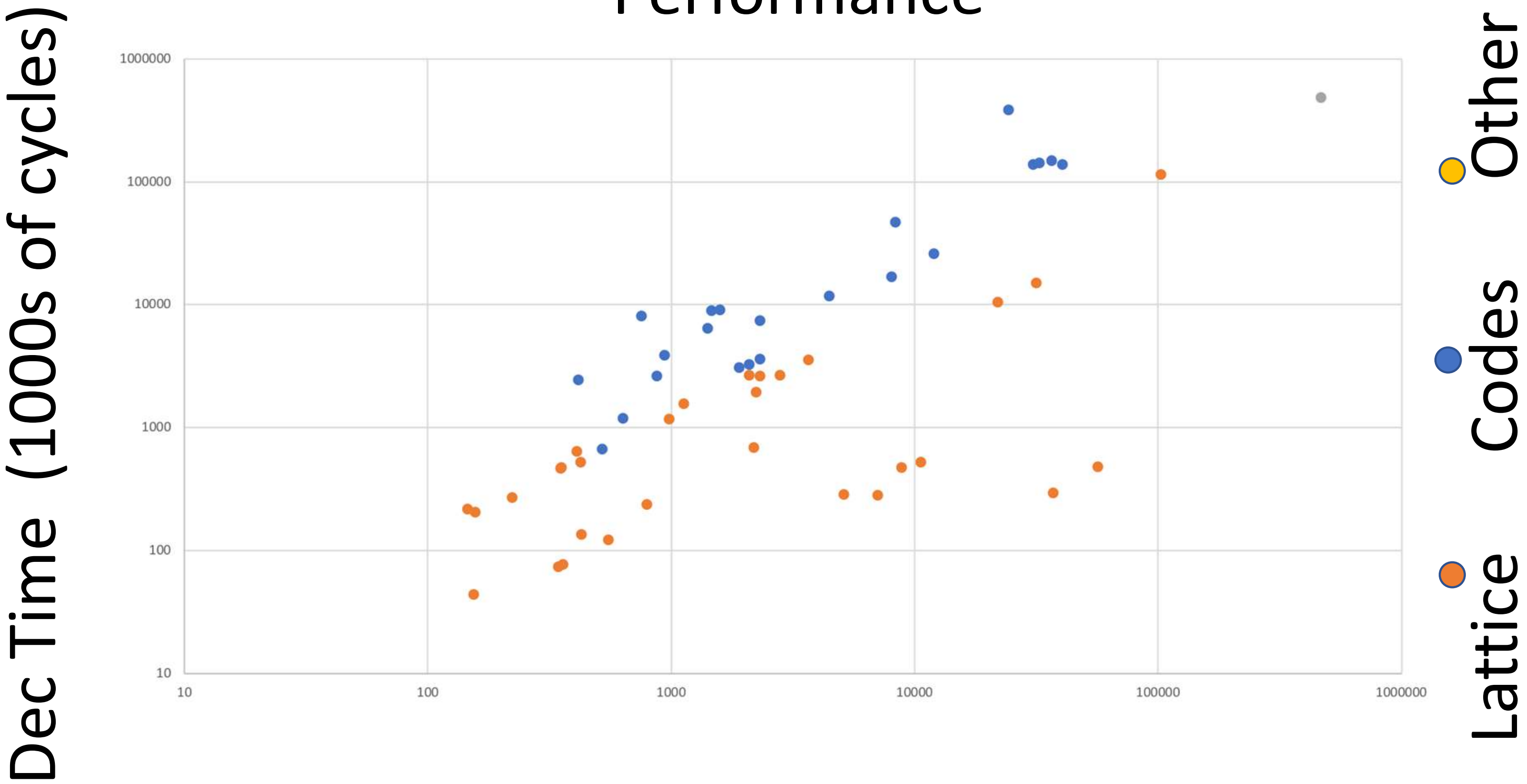
$$x = (xS)S^{-1} = \begin{pmatrix}1 & 0 & 0 & 0\end{pmatrix}\begin{pmatrix}1 & 1 & 0 & 1\\ 1 & 1 & 0 & 0\\ 0 & 1 & 1 & 1\\ 1 & 0 & 0 & 1\end{pmatrix} = \begin{pmatrix}1 & 1 & 0 & 1\end{pmatrix}$$

KEM/Encryption (Category 1)



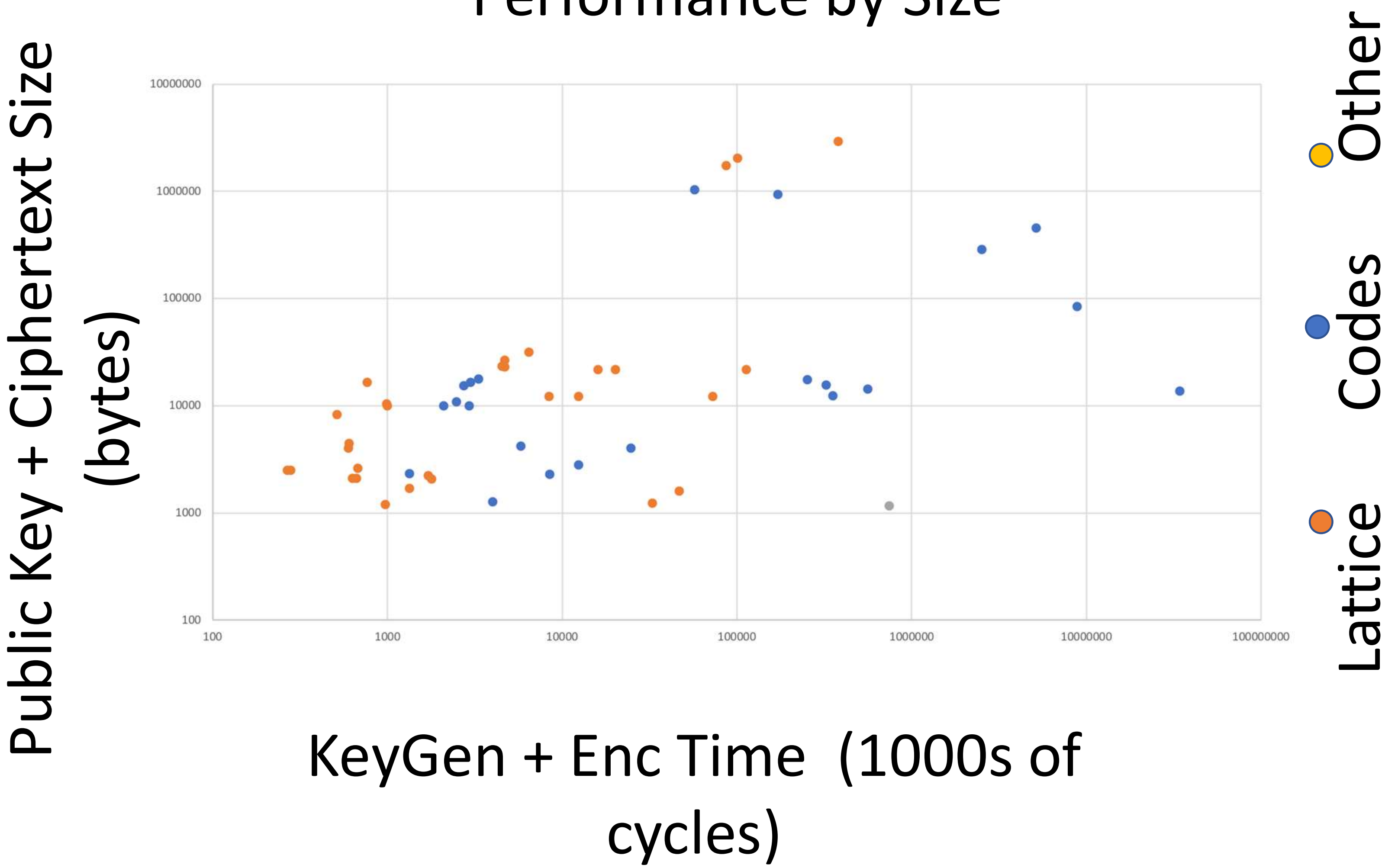
Hung-Jr, Shiu Ph. D

KEM/Encryption (Category 3) Performance



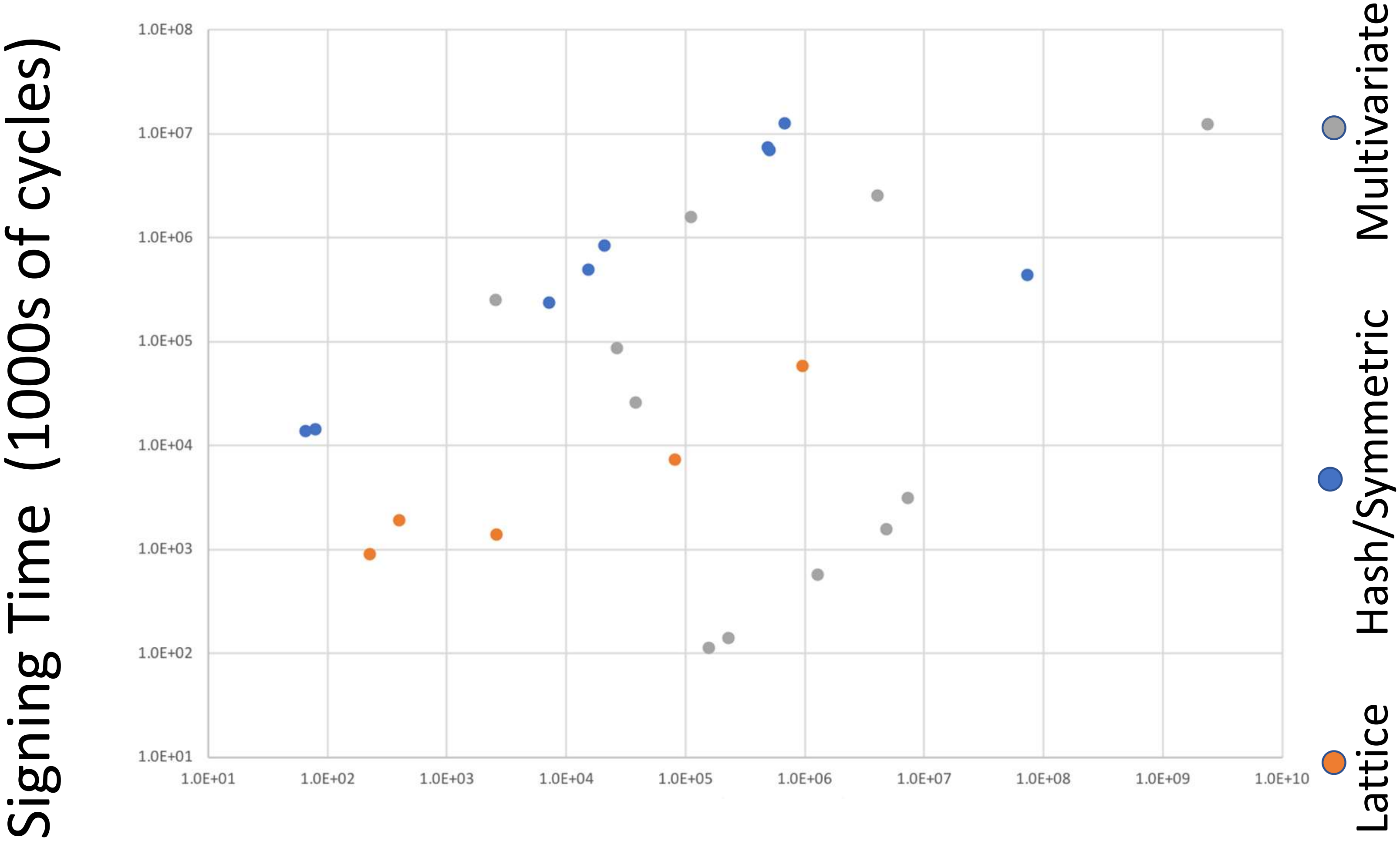
Hung-Jr, Shiu Ph. D

KEM/Encryption (Category 3) Performance by Size



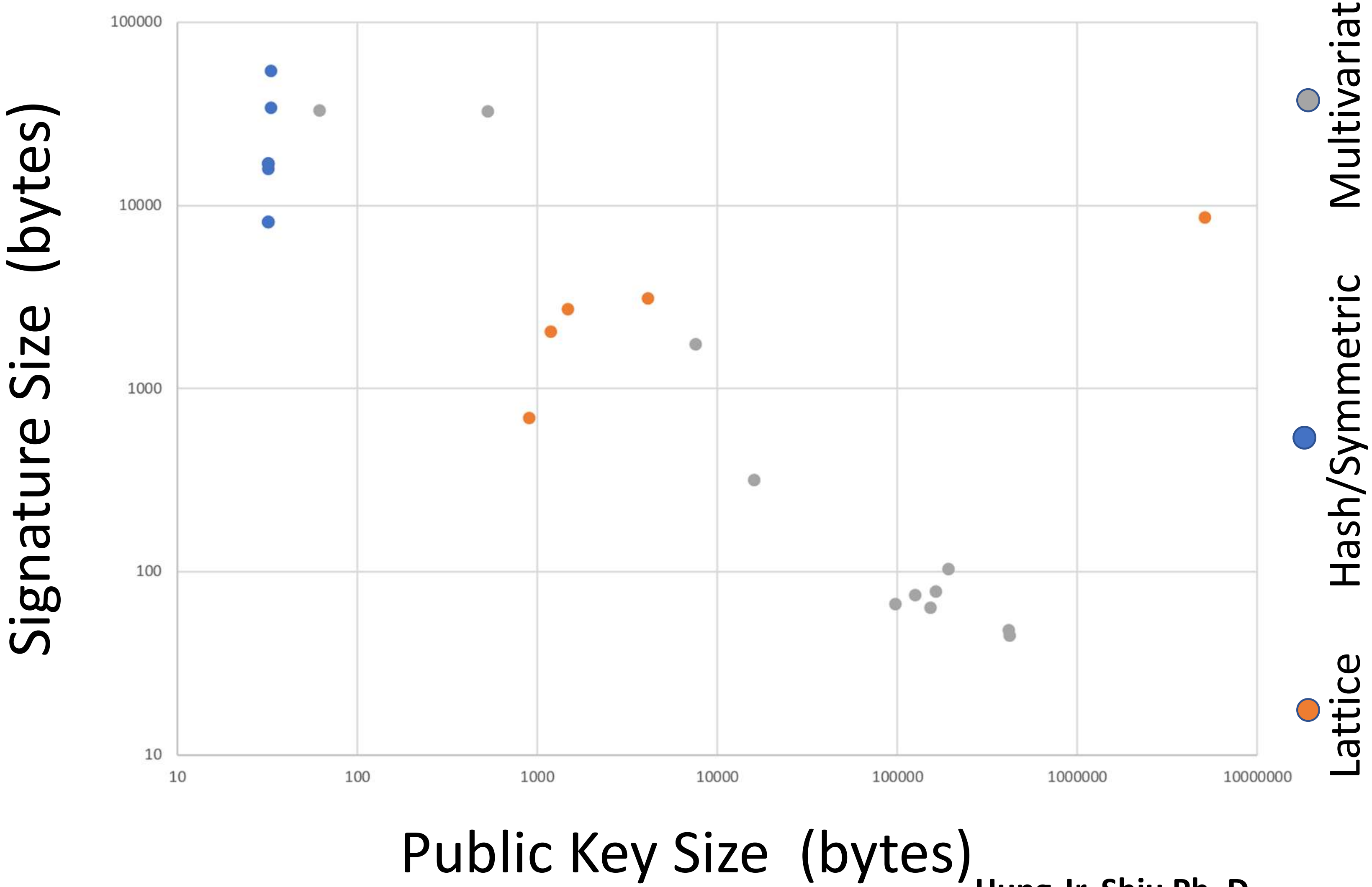
Hung-Jr, Shiu Ph. D

Signatures (Category 1) Performance



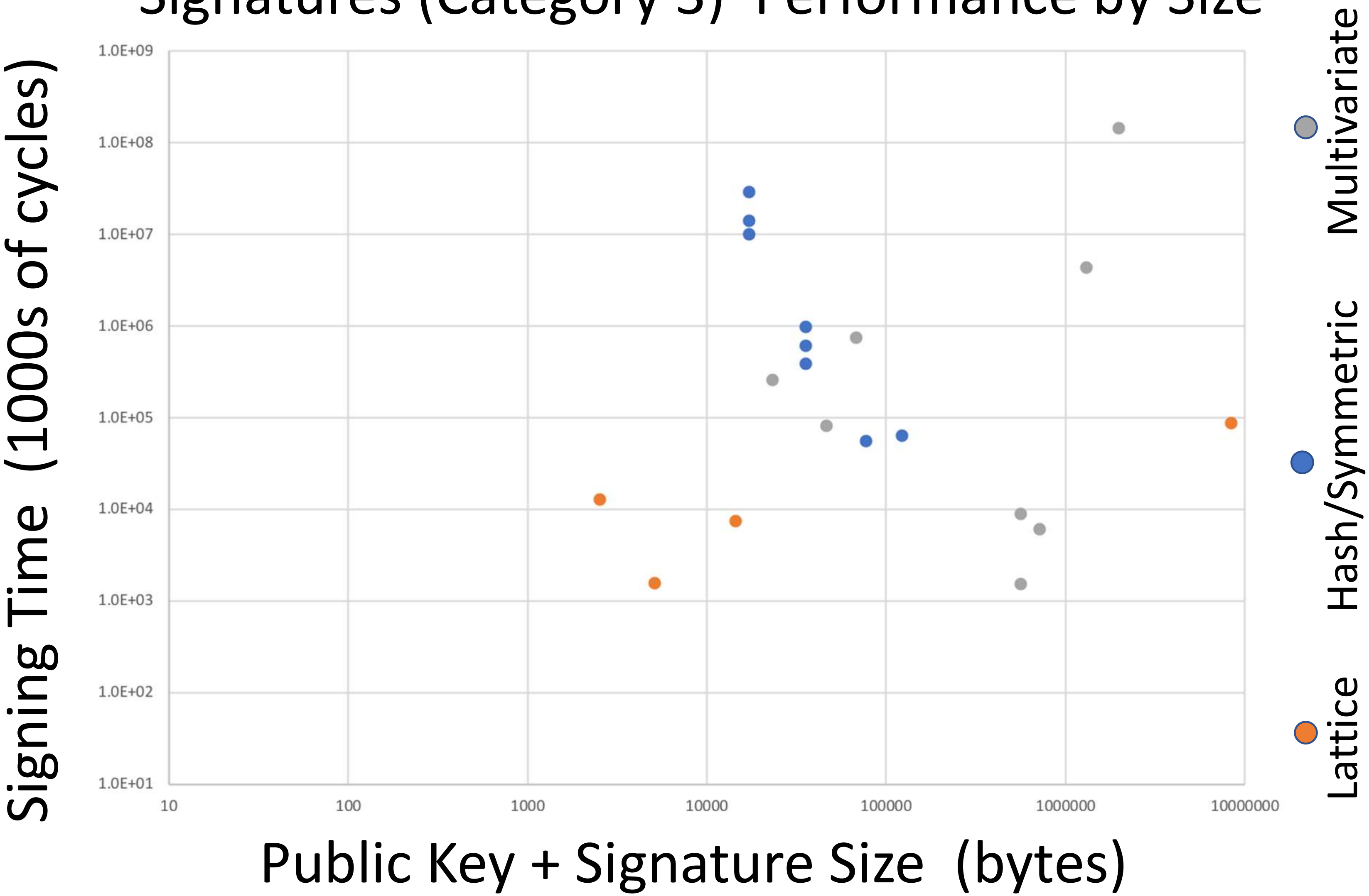
Hung-Jr, Shiu Ph. D

Signature (Category 1) Sizes



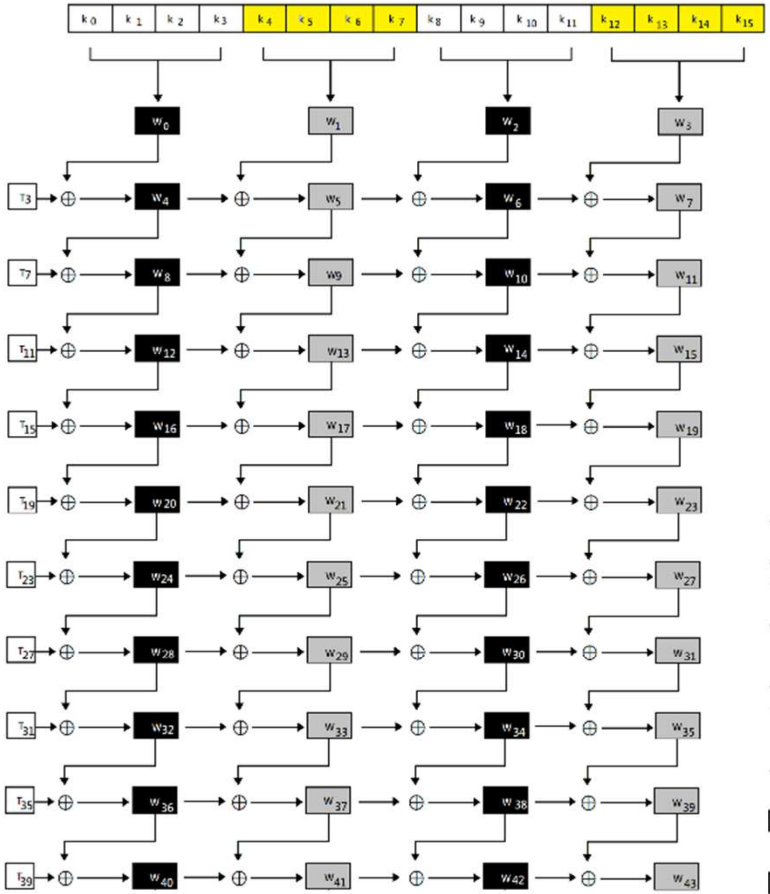
Hung-Jr, Shiu Ph. D

Signatures (Category 3) Performance by Size

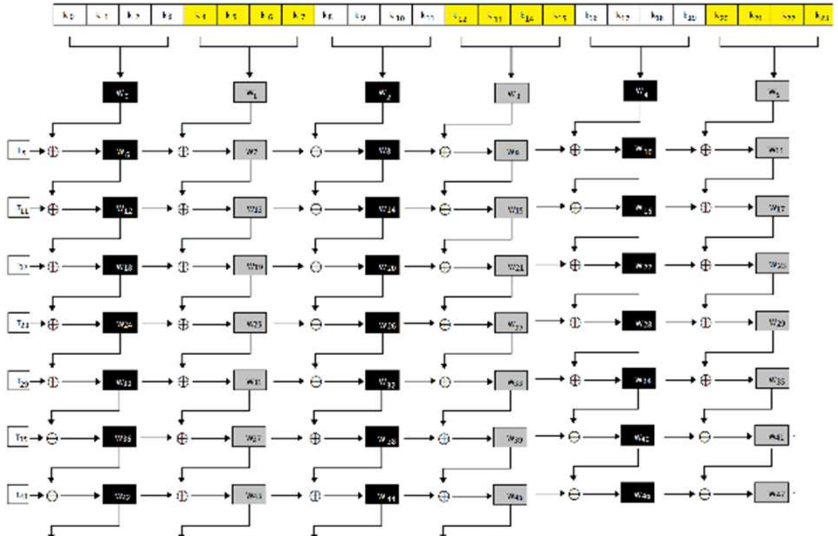


Hung-Jr, Shiu Ph. D

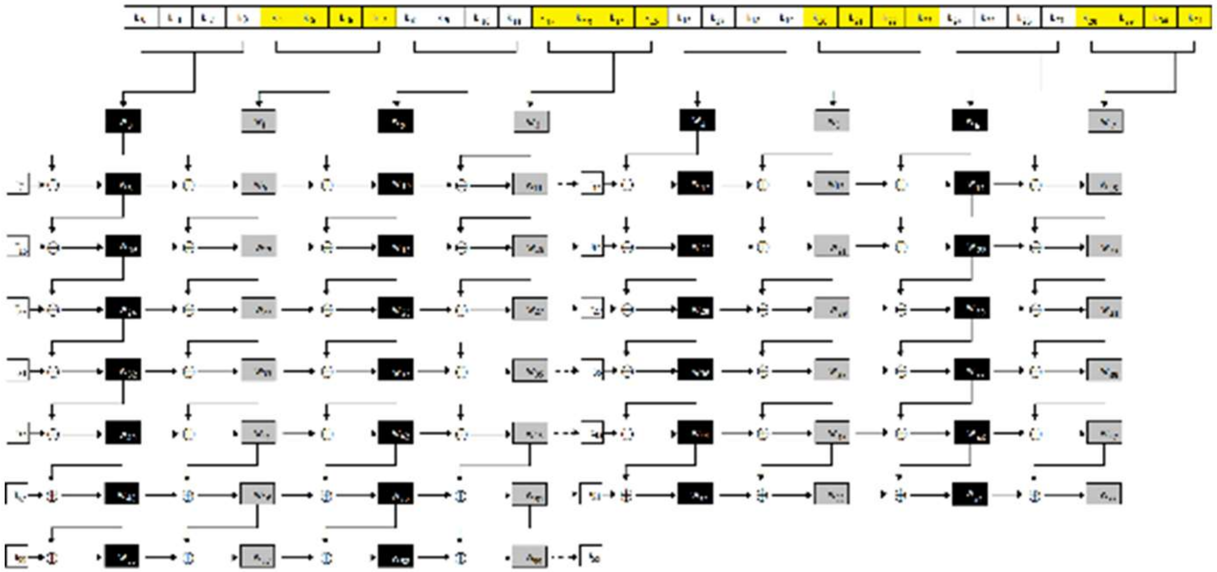
Recall AES key scheduling



Ti=Sub(Rot(Wi-1))+Rcom[i/4]

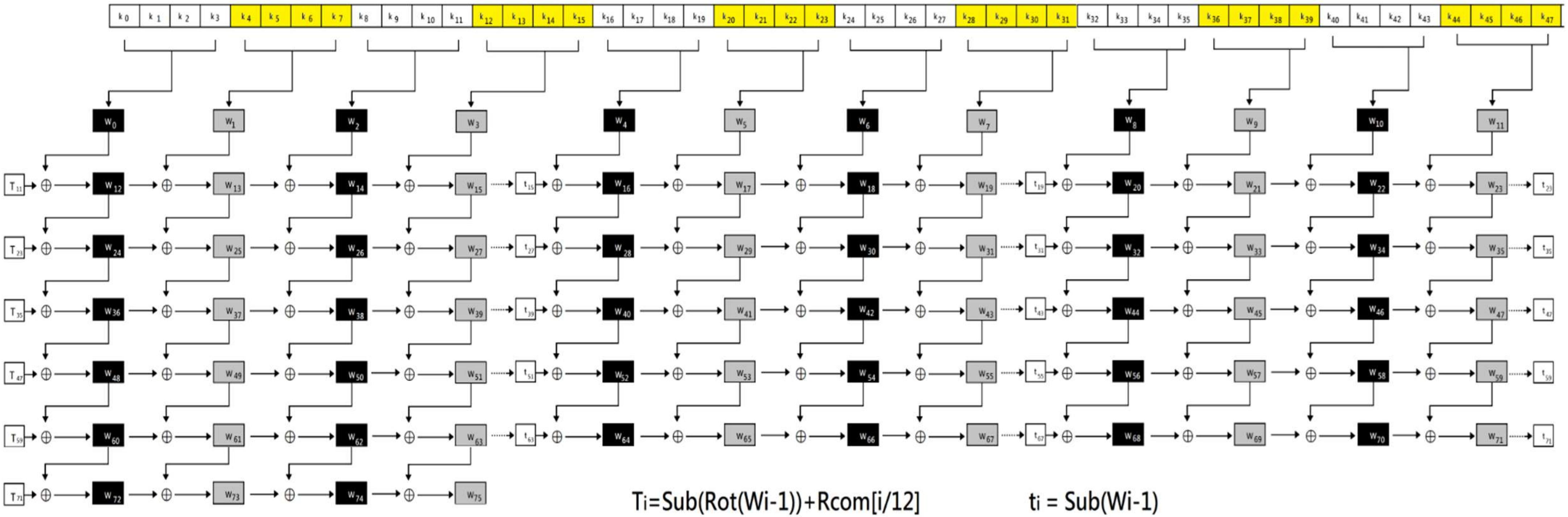


Ti =Sub(Rot(Wi-1))+Rcom[i/6]



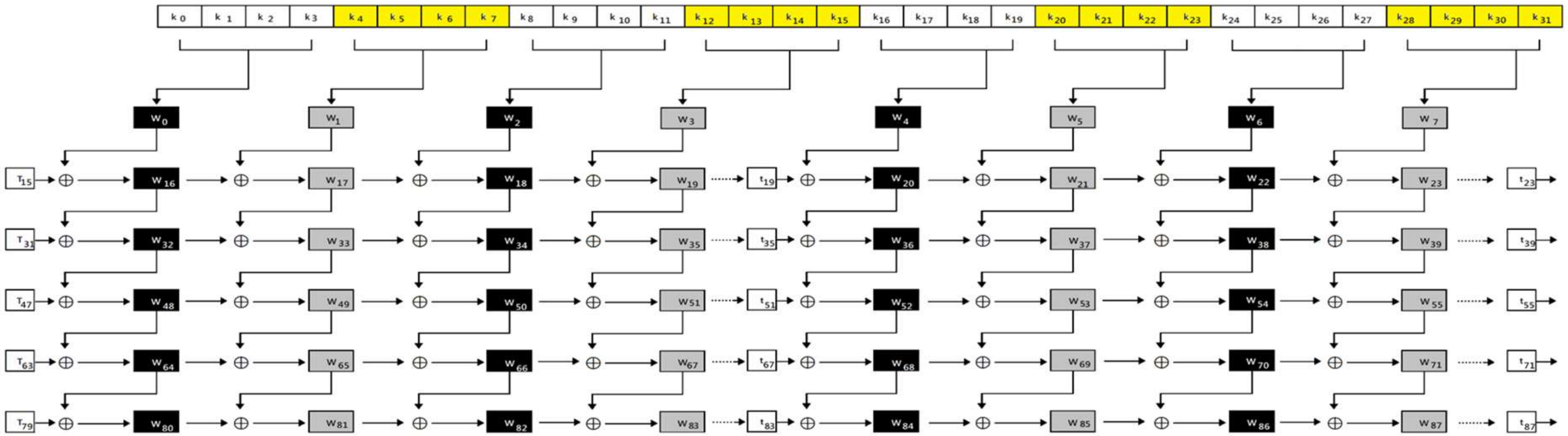
Ti=Sub(Rot(Wi-1))+Rcom[i/8] ti =Sub(Wi-1)

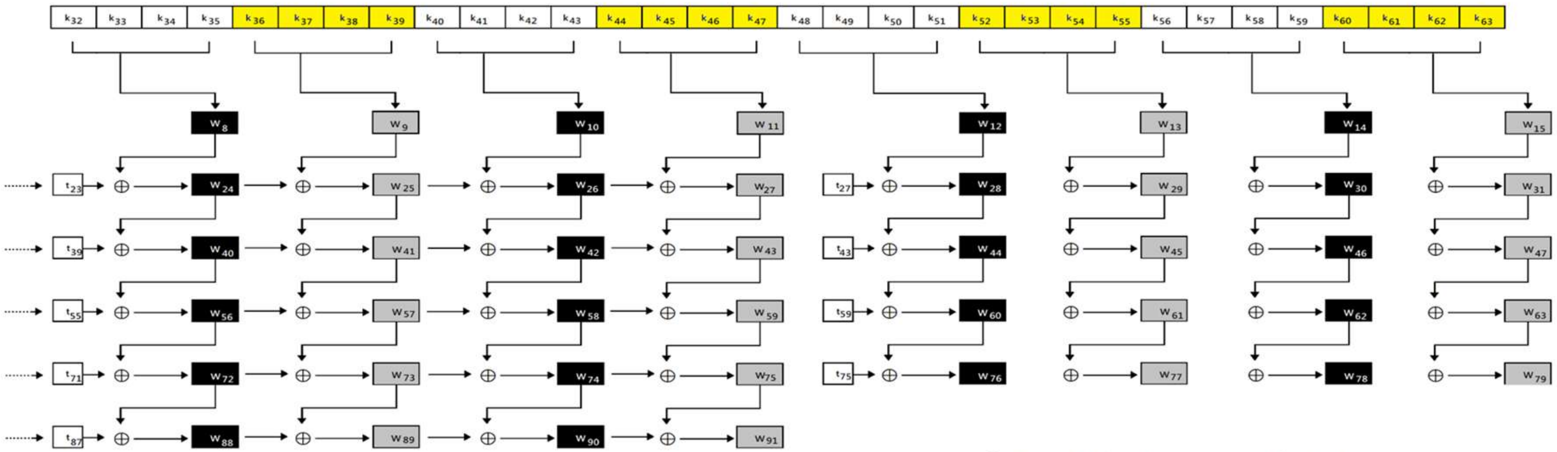
AES 384



$T_i = Sub(Rot(W_{i-1})) + Rcom[i/12]$ $\quad t_i = Sub(W_{i-1})$

AES 512





T_i=Sub(Rot(Wi-1))+Rcom[i/16]

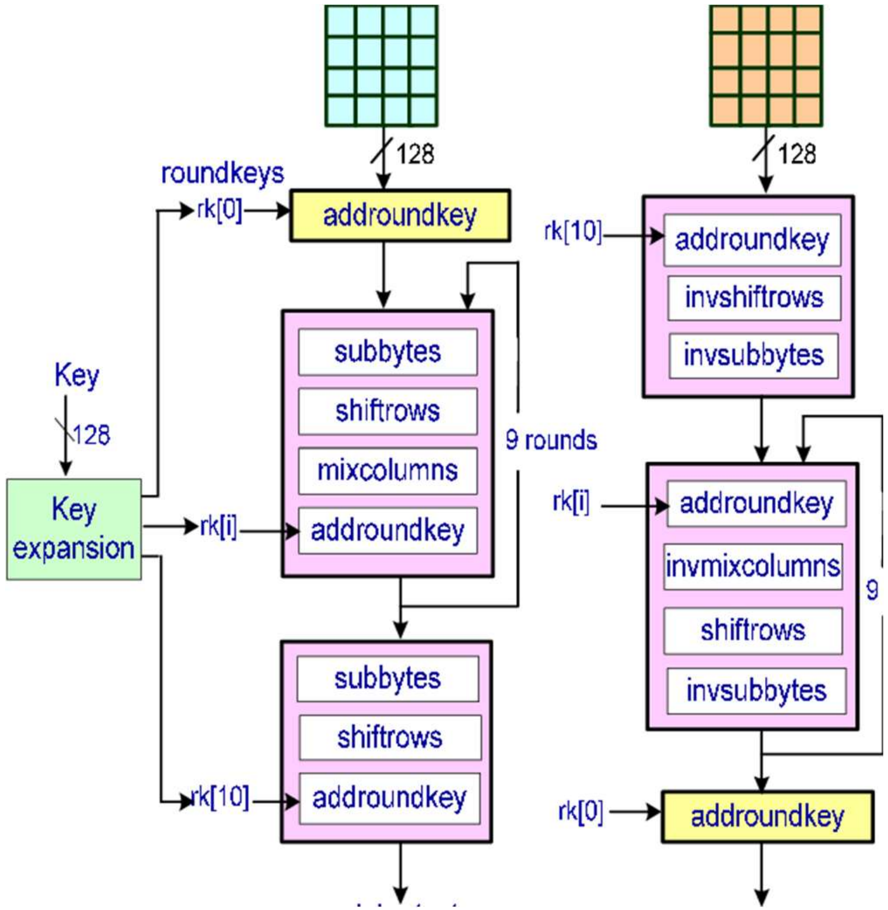
t_i =Sub(Wi-1)

Under Grover’s algortihm

- AES 384 -> level III (2^{192})
- AES 512 -> level V (2^{256})

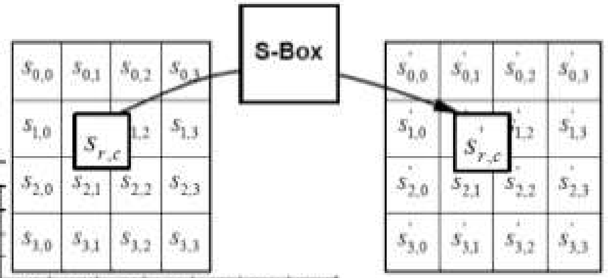
Customized AES

- Recall parameters:
  - Mixcolumns
  - Sbox



Subbytes

AES: SubBytes transformation

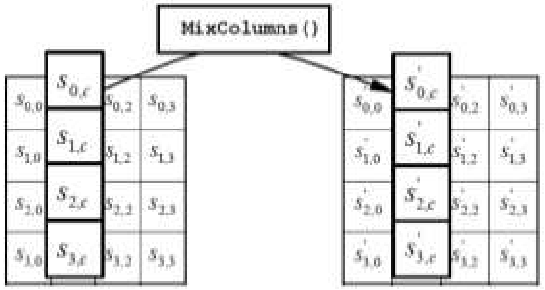


		y															
		0	1	2	3	4	5	6	7	8							
x	0	63	7c	77	7b	f2	6b	6f	c5	30							
	1	ca	82	c9	7d	fa	59	47	f0	ad							
	2	b7	fd	93	26	36	3f	f7	cc	34							
	3	04	c7	23	c3	18	96	05	9a	07	12	80	e2	eb	27	b2	75
	4	09	83	2c	1a	1b	6e	5a	a0	52	3b	d6	b3	29	e3	2f	84
	5	53	d1	00	ed	20	fc	b1	5b	6a	cb	be	39	4a	4c	58	cf
	6	d0	ef	aa	fb	43	4d	33	85	45	f9	02	7f	50	3c	9f	a8
	7	51	a3	40	8f	92	9d	38	f5	bc	b6	da	21	10	ff	f3	d2
	8	cd	0c	13	ec	5f	97	44	17	c4	a7	7e	3d	64	5d	19	73
	9	60	81	4f	dc	22	2a	90	88	46	ee	b8	14	de	5e	0b	db
	a	e0	32	3a	0a	49	06	24	5c	c2	d3	ac	62	91	95	e4	79
	b	e7	c8	37	6d	8d	d5	4e	a9	6c	56	f4	ea	65	7a	ae	08
	c	ba	78	25	2e	1c	a6	b4	c6	e8	dd	74	1f	4b	bd	8b	8a
	d	70	3e	b5	66	48	03	f6	0e	61	35	57	b9	86	c1	1d	9e
	e	e1	f8	98	11	69	d9	8e	94	9b	1e	87	e9	ce	55	28	df
	f	8c	a1	89	0d	bf	e6	42	68	41	99	2d	0f	b0	54	bb	16

Mixcolumns

AES: MixColumns transformation

$$\begin{bmatrix} s'_{0,c} \\ s'_{1,c} \\ s'_{2,c} \\ s'_{3,c} \end{bmatrix} = \begin{bmatrix} 02 & 03 & 01 & 01 \\ 01 & 02 & 03 & 01 \\ 01 & 01 & 02 & 03 \\ 03 & 01 & 01 & 02 \end{bmatrix} \begin{bmatrix} s_{0,c} \\ s_{1,c} \\ s_{2,c} \\ s_{3,c} \end{bmatrix}$$



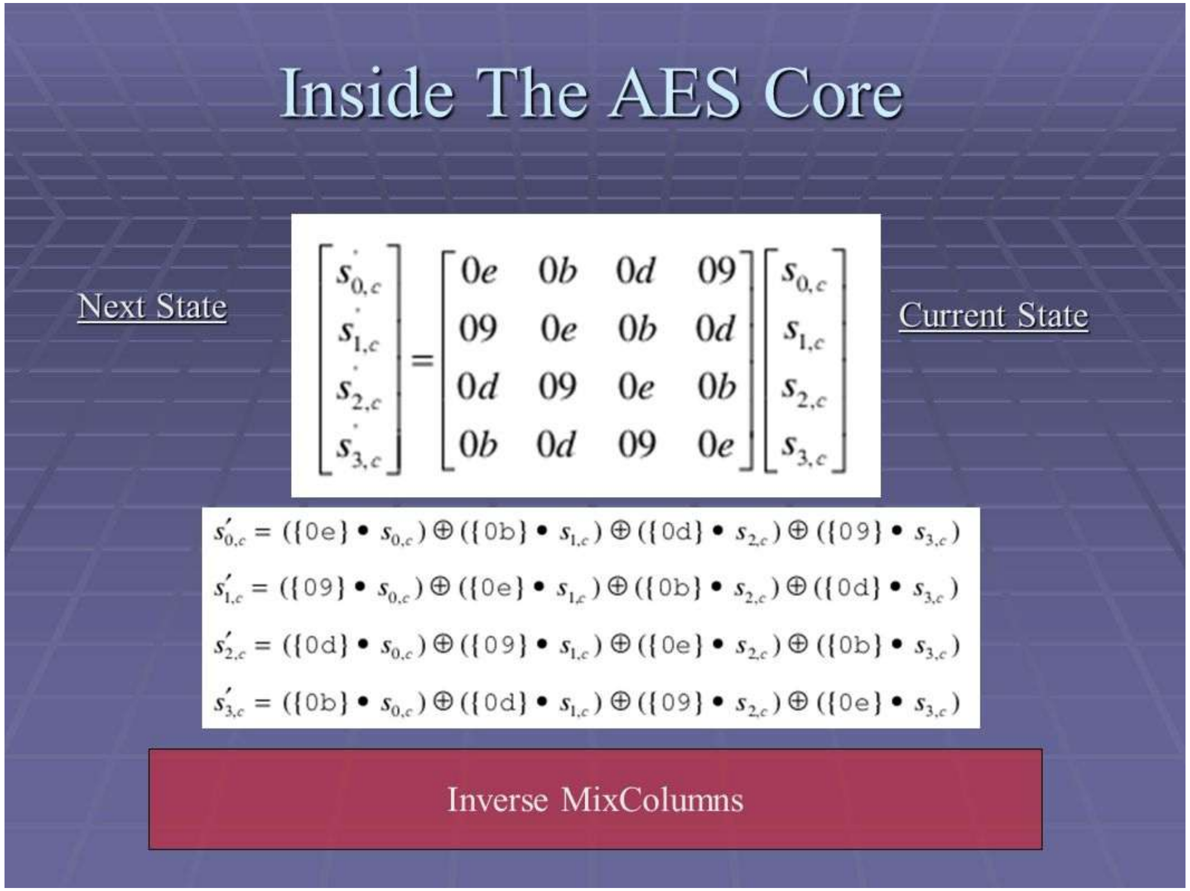
$$s'_{0,c} = (\{02\} \bullet s_{0,c}) \oplus (\{03\} \bullet s_{1,c}) \oplus s_{2,c} \oplus s_{3,c}$$

$$s'_{1,c} = s_{0,c} \oplus (\{02\} \bullet s_{1,c}) \oplus (\{03\} \bullet s_{2,c}) \oplus s_{3,c}$$

$$s'_{2,c} = s_{0,c} \oplus s_{1,c} \oplus (\{02\} \bullet s_{2,c}) \oplus (\{03\} \bullet s_{3,c})$$

$$s'_{3,c} = (\{03\} \bullet s_{0,c}) \oplus s_{1,c} \oplus s_{2,c} \oplus (\{02\} \bullet s_{3,c})$$

Inverse Mixcolumns



Mixcolumns and Inverse Mixcolumns

- All the addition and multiplication are done under a Finite Field ($GF(2^8)$).

$$\begin{bmatrix} 02 & 03 & 01 & 01 \\ 01 & 02 & 03 & 01 \\ 01 & 01 & 02 & 03 \\ 03 & 01 & 01 & 02 \end{bmatrix} \cdot \begin{bmatrix} 0E & 0B & 0D & 09 \\ 09 & 0E & 0B & 0D \\ 0D & 09 & 0E & 0B \\ 0B & 0D & 09 & 0E \end{bmatrix} = \begin{bmatrix} 01 & 00 & 00 & 00 \\ 00 & 01 & 00 & 00 \\ 00 & 00 & 01 & 00 \\ 00 & 00 & 00 & 01 \end{bmatrix}$$

Operations under GF(2^8)

- $a + b = a \oplus b\ (exlusive\ or)$

- $a \times b = \begin{Bmatrix} b = b_7 b_6 b_5 b_4 b_3 b_2 b_1 b_0\ for\ each\ b_i, \\ if\ a < 0x80, a = a \ll i \\ if\ a \geq 0x80, a = (a \ll i)\%0x11B \end{Bmatrix}$

Variant AES (1/2)

- Replace the Mixcolumns and find the Inverse Mixcolumns.
- Use Gaussian-Jordan Elimination to find the inverse matrix of Mixcolumns. (Linear Algebra)
- Notice that should be under GF(2^8).

Variant AES (2/2)

- Use new S-box, not the standard.
- Use Lagrange interpolation to evaluate the algebra structure, it should obtain equal or higher degree than standard S-box (255).
- Algebra structure of the standard S-box:

$$S_{RD}[x] = 05 \cdot x^{254} + 09 \cdot x^{253} + F9 \cdot x^{251} + 25 \cdot x^{247} + F4 \cdot x^{239} + 01 \cdot x^{223} + B5 \cdot x^{191} + 8F \cdot x^{127} + 63$$

Lagrange Interpolation

Given a set of data $(x_1, y_1), (x_2, y_2), \ldots, (x_n, y_n)$ where $x_1 \neq x_2 \neq \ldots \neq x_n$

$$y \approx \frac{(x-x_2)(x-x_3)\ldots(x-x_n)}{(x_1-x_2)(x_1-x_3)\ldots(x_1-x_n)} y_1 + \frac{(x-x_1)(x-x_3)\ldots(x-x_n)}{(x_2-x_1)(x_2-x_3)\ldots(x_2-x_n)} y_2 +$$

$$\ldots + \frac{(x-x_1)(x-x_2)\ldots(x-x_{n-1})}{(x_n-x_1)(x_n-x_2)\ldots(x_n-x_{n-1})} y_n$$

$$= \sum_{k=1}^{n} \prod_{\substack{j=1 \\ j \neq k}}^{n} \frac{(x-x_j)}{(x_k-x_j)} y_k$$

Given $x_0 = 1, f(x_0) = 1;\ x_1 = 2, f(x_1) = 8;\ x_2 = 3, f(x_2) = 27$

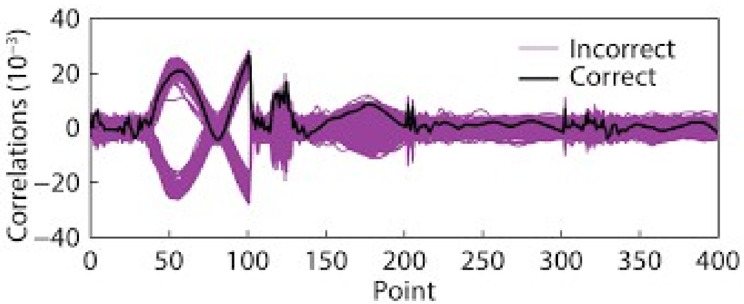
$$L(x) = 1 \cdot \frac{x-2}{1-2} \cdot \frac{x-3}{1-3} + 8 \cdot \frac{x-1}{2-1} \cdot \frac{x-3}{2-3} + 27 \cdot \frac{x-1}{3-1} \cdot \frac{x-2}{3-2} = 6x^2 - 11x + 6$$

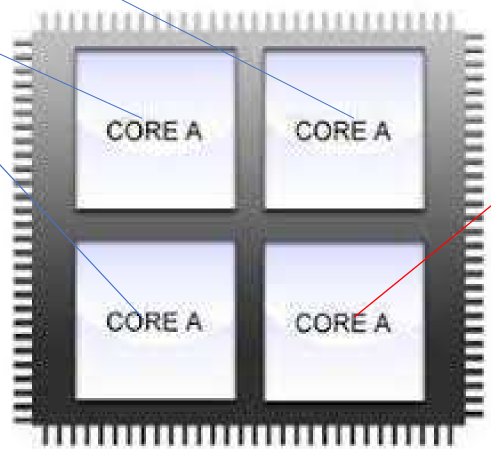
Signal Masking -Information Hiding (Steganography)

- Use a core to run high frequency operations, it will generate higher temperature or noise to mask real encryptions.

Run AES

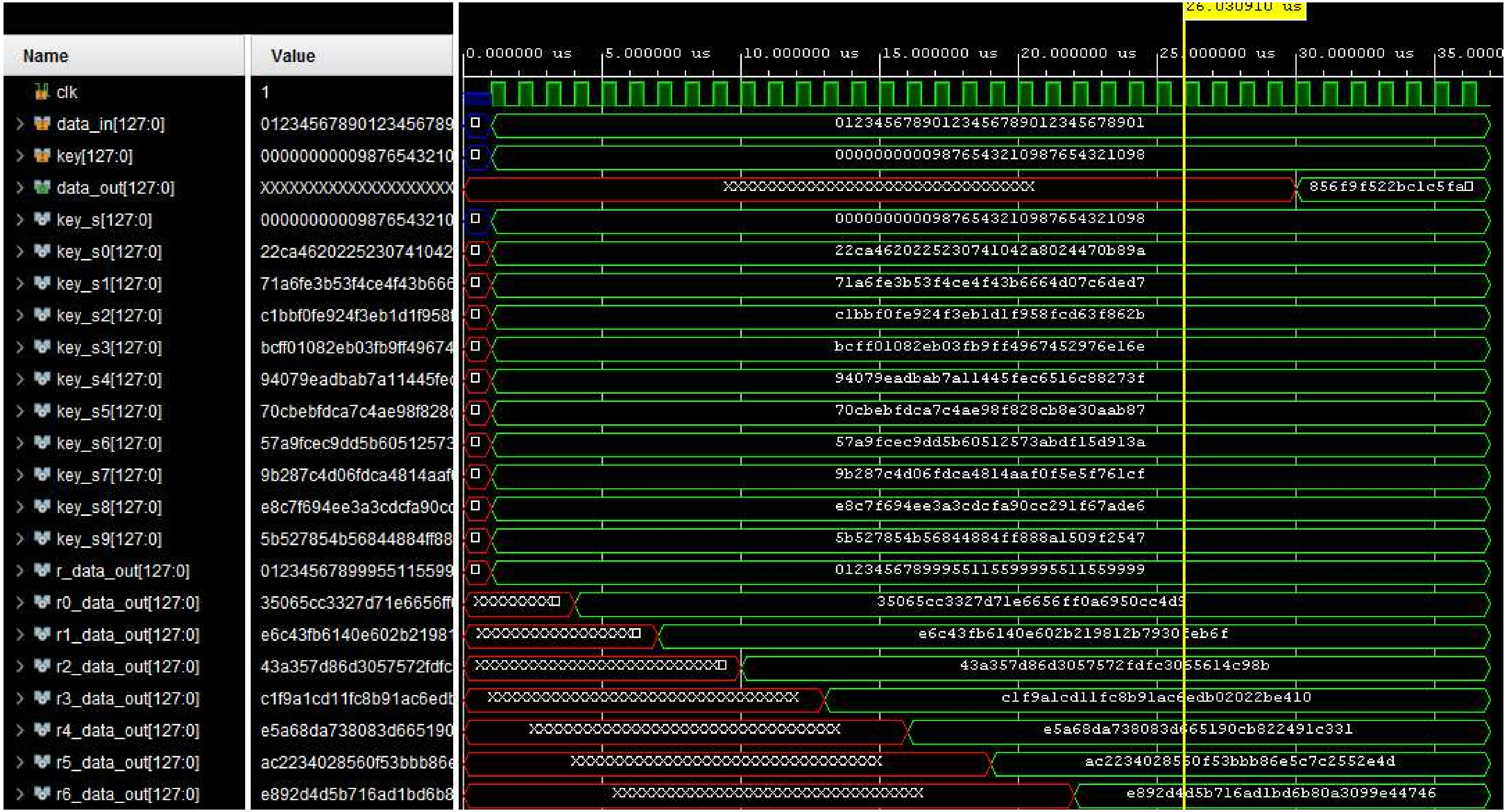
Run high frequency operations





Hung-Jr, Shiu Ph. D

Hardware pipeline



Conclusions

- For symmetric encryption, to against SCA,
  - IoT is a big threats of crypto-module, a slight modification of standard.
  - Use multicore hardware and steganography strategy.

- For asymmetric encryption, adopt PQC and implement on FPGA to speed up.

Thanks for your attention

Q