# Post-quantum Key Exchange from LWE

Jintai Ding, University of Cincinnati

Worcester Polytechnic Institute, September 2, 2016

# The context of our work - PKC

- Diffie-Hellman's revolution idea – Public key cryptography
  Symmetric systems versus Asymmetric systems
- The work of RSA – the critical role of mathematics
- The Internet and the PKCs
  Internet can not work without PKCs.

# The context of our work - PQC

- Shor's quantum algorithm
- Post-quantum cryptography
  Develop public key cryptosystems that could resist future quantum computer attacks

# The Preparation for the Future

- The first Quantum-Safe-Crypto Workshop

  26 - 27 September, 2013

  **ETSI – the European Telecommunications Standards Institute** at SOPHIA ANTIPOLIS, FRANCE
- The second Quantum-Safe-Crypto Workshop

  6 - 2 October , 2014, Ottawa, Canada
  White paper
- The Quantum-Safe-Crypto Workshop at **NIST: National Institute of Standard of Technology**,

  April 7-8, 2015, Washington DC

# A slide of M. MOSCA

12

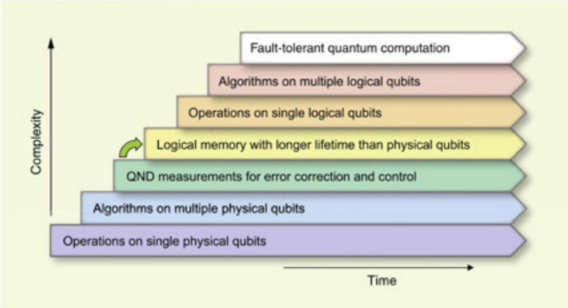



**Fig. 1.** Seven stages in the development of quantum information processing. Each advancement requires mastery of the preceding stages, but each also represents a continuing task that must be perfected in parallel with the others. Superconducting qubits are the only solid-state implementation at the third stage, and they now aim at reaching the fourth stage (green arrow). In the domain of atomic physics and quantum optics, the third stage had been previously attained by trapped ions and by Rydberg atoms. No implementation has yet reached the fourth stage, where a logical qubit can be stored, via error correction, for a time substantially longer than the decoherence time of its physical qubit components.

# A commercial for PQC from NSA

# What do we really need ?– a slides of L. Chen from NIST

## Practical Challenge

- Quantum computing will break many public-key cryptographic algorithms/schemes
  - Key agreement (e.g. DH and MQV)
  - Digital signatures (e.g. RSA and DSA)
  - Encryption (e.g. RSA)
- These algorithms have been used to protect Internet protocols (e.g. IPsec) and applications (e.g.TLS)
- NIST is studying "quantum-safe" replacements

# The call from NIST

In PQC2016 in Japan, NIST make a call for quantum resistant algorithms by Dustin Moody

*Deadline: November 2017*

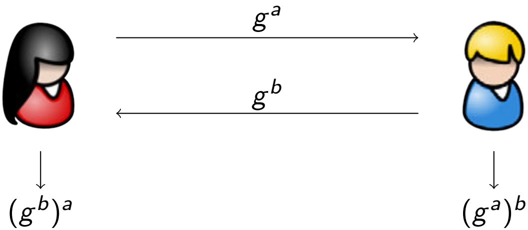
# Post Quantum Needs – Functionality

- Key Exchange – for secure communications
- Signatures – for Authentication

# Key Exchange Applications — SSL/TLS

- RSA
- Diffie–Hellman
- Our goal – replacements for post quantum world

# Diffie-Hellman Key Exchange



# Generalizing DH

- DH works because maps $f(x) = x^a$ and $h(x) = x^b$ commute

$$f \circ h = h \circ f,$$

  $\circ$ – composition
  **Nonlinearity**

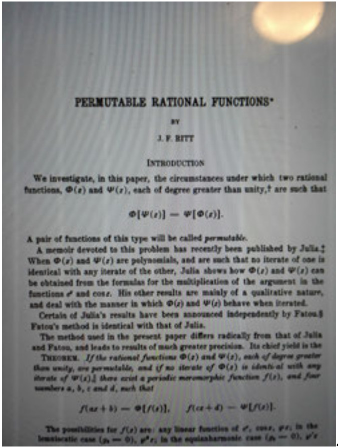
- Many attempts – Braid group etc

# Generalizing DH

- When do we have commuting *nonlinear* maps?
  - Powers of $x$ (normal DH)
  - Iterates of a polynomial
  - Julia (Fatou)– Mmoire sur la permutabilit des fractions rationnelles, Annales de l'Ecole Normale Suprieure, vol. 39 (1922), pp. 131-215.
  - J. Ritt (1923) – Power polynomials, Chebyshev polynomials. Elliptic curve

# Who is J. Ritt: 1893-1951

# Who is J. Ritt: 1923: PERMUTABLE RATIONAL FUNCTIONS



PERMUTABLE RATIONAL FUNCTIONS*

BY

J. F. RITT

INTRODUCTION

We investigate, in this paper, the circumstances under which two rational functions, $\Phi(z)$ and $\Psi(z)$, each of degree greater than unity,† are such that

$$\Phi[\Psi(z)] = \Psi[\Phi(z)].$$

A pair of functions of this type will be called *permutable*.

A memoir devoted to this problem has recently been published by Julia.‡ When $\Phi(z)$ and $\Psi(z)$ are polynomials, and are such that no iterate of one is identical with any iterate of the other, Julia shows how $\Phi(z)$ and $\Psi(z)$ can be obtained from the formulas for the multiplication of the argument in the functions $e^z$ and $\cos z$. His other results are mainly of a qualitative nature, and deal with the manner in which $\Phi(z)$ and $\Psi(z)$ behave when iterated.

Certain of Julia's results have been announced independently by Fatou.§ Fatou's method is identical with that of Julia.

The method used in the present paper differs radically from that of Julia and Fatou, and leads to results of much greater precision. Its chief yield is the

THEOREM. *If the rational functions $\Phi(z)$ and $\Psi(z)$, each of degree greater than unity, are permutable, and if no iterate of $\Phi(z)$ is identical with any iterate of $\Psi(z)$,‖ there exist a periodic meromorphic function $f(z)$, and four numbers $a$, $b$, $c$ and $d$, such that*

$$f(az+b) = \Phi[f(z)], \qquad f(cz+d) = \Psi[f(z)].$$

The possibilities for $f(z)$ are: any linear function of $e^z$, $\cos z$, $\wp z$; in the lemniscatic case ($g_3 = 0$), $\wp^2 z$; in the equianharmonic case ($g_2 = 0$), $\wp' z$

J. Ritt (1923) – Power polynomials, Chebychev polynomials. Elleptic curve

# Generalizing DH

Our basic idea — adding "small" **noise or perturbation**:

- (Ring) LWE approximately commutes—use to build DH generalization
  From
  $$(s_1 \times a) \times s_2 = s_1 \times (a \times s_2)$$
  to
  $$(as_1 + e_1)s_2 \approx s_1 as_2 \approx (as_2 + e_2)s_1.$$

# A historical Note

Our basic idea — adding "small" **noise or perturbation** is not new!!!

- GCHQ – Communications-Electronics Security Group(CESG) – James Elias – "Invention of non-secret encryption" 1969 Clifford Cocks – RSA, Malcolm Williamson – DH, 1973
- The forgotten inspiration of J. Elias –
  "Ellis said that the idea first occurred to him after reading a paper from World War II by someone at Bell Labs describing a way to protect voice communications by the receiver adding (and then later subtracting) random noise (possibly this 1944 paper[4] or the 1945 paper co-authored by Claude Shannon)"
  – Wikipedia

# Learning with Errors [2006, Regev]

$$\underbrace{\begin{pmatrix} b_1 \\ b_2 \\ \vdots \\ b_m \end{pmatrix}}_{\vec{b}} = \underbrace{\begin{pmatrix} a_{11} & a_{12} & \dots & a_{1n} \\ a_{21} & a_{22} & \dots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{m1} & a_{m2} & \dots & a_{mn} \end{pmatrix}}_{A} \underbrace{\begin{pmatrix} s_1 \\ s_2 \\ \vdots \\ s_n \end{pmatrix}}_{\vec{s}} + \underbrace{\begin{pmatrix} e_1 \\ e_2 \\ \vdots \\ e_m \end{pmatrix}}_{\vec{e}}$$

- Approximate system over $\mathbb{Z}_q$
- Hard to find $\vec{s}$ from $A, \vec{b}$.
- Hard to tell if $\vec{s}$ even exists
- Reduction to lattice approximation problems

# Ring LWE

**Definition**

Let $n$ be a power of 2, $q \equiv 1 \pmod{2n}$ prime. Define the ring

$$R_q = \frac{\mathbb{Z}_q[x]}{(x^n + 1)}.$$

- Again, $b = as + e$ hard to find $s$
- Hard to distinguish from uniform $b$
- Approximation problems on *ideal* lattices
- More efficient than standard LWE

# Diffie-Hellman from Ideal Lattices

$$\xrightarrow{p_A = as_A + 2e_A}$$

$$\xleftarrow{p_B = as_B + 2e_B}$$

- Public $a \in R_q$. Acts like generator $g$ in DH.

# Diffie-Hellman from Ideal Lattices

$$\xrightarrow{p_A = as_A + 2e_A}$$

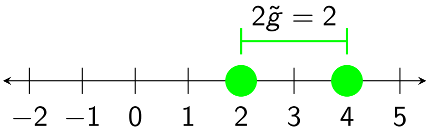
$$\xleftarrow{p_B = as_B + 2e_B}$$

$$\downarrow \qquad\qquad\qquad\qquad \downarrow$$

$$k_A = s_A p_B = aS_AS_B + 2S_Ae_B \quad \approx \quad k_B = p_A s_B = aS_AS_B + 2S_Be_A$$

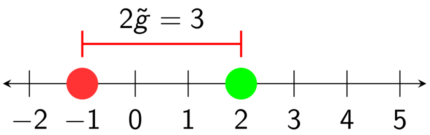
- Public $a \in R_q$. Acts like generator $g$ in DH.
- Each side's key is only *approximately* equal to the other.
- Difference is even—same low bits.
- No authentication—MitM

# Wrap-around Illustrated




- Difference 2, both even.

# Wrap-around Illustrated




- Difference 2, both even.
- But wait! If $q = 5$, $\mathbb{Z}_q = \{-2, -1, 0, 1, 2\}$.
- 4 becomes $-1$, now parities disagree!

# Compensating for Wrap-Around

- $g = 2S_A e_B - 2S_B e_A$.
- Recall: $|g^{(j)}| < \frac{q}{8}$
- Define $E = \{-\lfloor \frac{q}{4} \rfloor, \ldots, \lfloor \frac{q}{4} \rceil\}$. Middle half of $\mathbb{Z}_q$.
- If $k_B^{(j)} \in E$, no wrap-around occurs; $k_A^{(j)} \equiv k_B^{(j)}$.
- If $k_B^{(j)} \notin E$, then $k_B^{(j)} + \frac{q-1}{2} \in E$
- If $k_B^{(j)} \notin E$, $k_A^{(j)} + \frac{q-1}{2} \equiv k_B^{(j)} + \frac{q-1}{2}$.

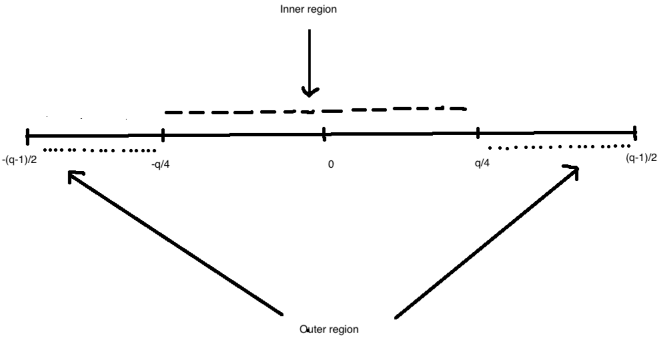
# Wrap-around Defeated

Define $w_B^{(j)} = \begin{cases} 0 & k_B^{(j)} \in E, \\ 1 & k_B^{(j)} \notin E. \end{cases}$ Then $k_B^{(j)} + w_B^{(j)} \frac{q-1}{2} \in E$.

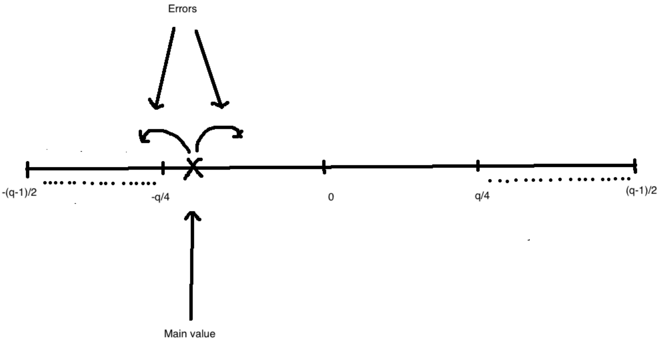
Also, $k_B^{(j)} + w_B^{(j)} \frac{q-1}{2} \equiv k_A^{(j)} + w_B^{(j)} \frac{q-1}{2} \pmod 2$.

- $k_B^{(j)} + w_B^{(j)} \frac{q-1}{2} \bmod q \ \bmod 2 = k_A^{(j)} + w_B^{(j)} \frac{q-1}{2} \bmod q \bmod 2$.
- Wrap-around correction $w_B = (w_B^{(0)}, w_B^{(1)}, \ldots, w_B^{(n-1)})$
- $\sigma_B = k_B + w_B \frac{q-1}{2} \bmod 2$.
- $\sigma_A = k_A + w_B \frac{q-1}{2} \bmod 2$.

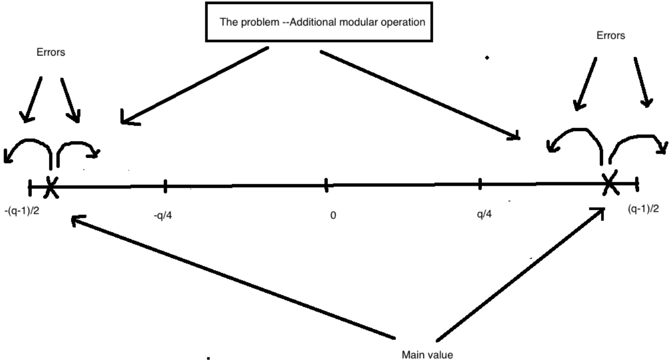
# Rounding Intuition – Region Division



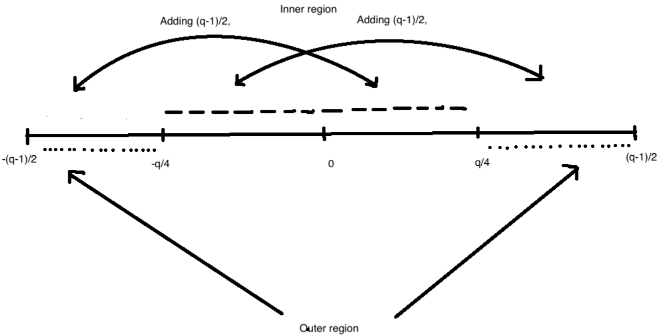
# Rounding Intuition – Inner Region



# Rounding Intuition – Outer Region problem



# Rounding Intuition




By adding (q-1)/2, two regions swap.

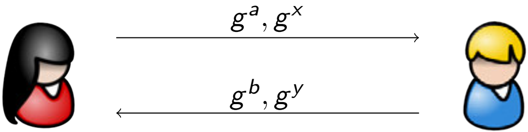
# Authentication: HMQV – To Resist Man-in-the-middle Attack and Achieve Forward Security

$$\xrightarrow{\quad g^a \quad}$$

$$\xleftarrow{\quad g^b \quad}$$

- Static keys $a$, $b$; tied to each party's identity.

# Authentication: HMQV – To Resist Man-in-the-middle Attack and Achieve Forward Security




- Static keys $a$, $b$; tied to each party's identity.
- Ephemeral keys $x$, $y$: **forward security**.

# Authentication: HMQV – To Resist Man-in-the-middle Attack and Achieve Forward Security

$$\xrightarrow{\quad g^a,\ g^x \quad}$$

$$\xleftarrow{\quad g^b,\ g^y \quad}$$

$$\downarrow \qquad\qquad\qquad\qquad \downarrow$$

$$\underbrace{\left(g^y (g^b)^e\right)^{x+da}}_{\sigma_A} \qquad\qquad \underbrace{\left(g^x (g^a)^d\right)^{y+eb}}_{\sigma_B}$$

- Static keys $a$, $b$; tied to each party's identity.
- Ephemeral keys $x$, $y$: **forward security**.
- Publicly derivable computations $d$, $e$.

# Authentication: HMQV – To Resist Man-in-the-middle Attack and Achieve Forward Security

$$\xrightarrow{\quad g^a, g^x \quad}$$

$$\xleftarrow{\quad g^b, g^y \quad}$$

$$\underbrace{(g^y (g^b)^e)^{x+da}}_{\sigma_A} = g^{(y+eb)(x+da)} = \underbrace{(g^x (g^a)^d)^{y+eb}}_{\sigma_B}$$

- Static keys $a$, $b$; tied to each party's identity.
- Ephemeral keys $x$, $y$: **forward security**.
- Publicly derivable computations $d$, $e$.
- Shared key is $K = H(\sigma_A) = H(\sigma_B) = H(g^{(y+be)(x+da)})$

# HMQV from Ideal Lattices

$$\xrightarrow{\quad p_A = as_A + 2e_A \quad}$$

$$\xleftarrow{\quad p_B = as_B + 2e_B \quad}$$

- $p_A, p_B$ as above. Public, static keys for authentication

# HMQV from Ideal Lattices

$$\xrightarrow{p_A = as_A + 2e_A,\, x_A = ar_A + 2f_A}$$

$$\xleftarrow{p_B = as_B + 2e_B,\, y_B = ar_B + 2f_B}$$

- $p_A, p_B$ as above. Public, static keys for authentication
- $x_A, y_B$ same form. Forward secrecy.

# HMQV from Ideal Lattices

$$\xrightarrow{p_A = as_A + 2e_A,\ x_A = ar_A + 2f_A}$$

$$\xleftarrow{p_B = as_B + 2e_B,\ y_B = ar_B + 2f_B}$$

$$\begin{aligned} k_A &= (p_B d + y_B)(s_A c + r_A) \\ &\quad +2dg_A \\ &\approx (aS_B d + ar_B)(s_A c + r_A) \end{aligned}$$

$$\begin{aligned} k_B &= (p_A c + x_A)(s_B d + r_B) \\ &\quad +2cg_B \\ &\approx (aS_A c + ar_A)(s_B d + r_B) \end{aligned}$$

- $p_A, p_B$ as above. Public, static keys for authentication
- $x_A, y_B$ same form. Forward secrecy.
- $c, d$ publicly derivable; $g_A, g_B$ random, small.

# Key Derivation

Obtaining shared secret from approximate shared secret:

$$\begin{aligned} k_A &= (k_A^{(0)}, k_A^{(1)}, \ldots, k_A^{(n-1)}) \\ k_B &= (k_B^{(0)}, k_B^{(1)}, \ldots, k_B^{(n-1)}) \\ \tilde{g} &= (g^{(0)}, g^{(1)}, \ldots, g^{(n-1)}) \\ k_A - k_B &= 2\tilde{g} \\ k_A &\equiv k_B \pmod 2 \end{aligned}$$

# Key Derivation

Obtaining shared secret from approximate shared secret:

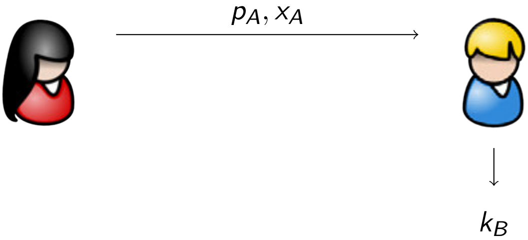
$$\begin{aligned} k_A &= (k_A^{(0)}, k_A^{(1)}, \ldots, k_A^{(n-1)}) \\ k_B &= (k_B^{(0)}, k_B^{(1)}, \ldots, k_B^{(n-1)}) \\ \tilde{g} &= (g^{(0)}, g^{(1)}, \ldots, g^{(n-1)}) \\ k_A - k_B &= 2\tilde{g} \\ k_A &\equiv k_B \pmod 2 \end{aligned}$$

- Each $k_A^{(j)} = k_B^{(j)} + 2g^{(j)}$.
- Each $g^{(j)}$ is small ($|g^{(j)}| < \frac{q}{8}$).
- Matching coefficients differ by small multiple of 2
- Take each coefficient mod 2, get $n$ bit secret

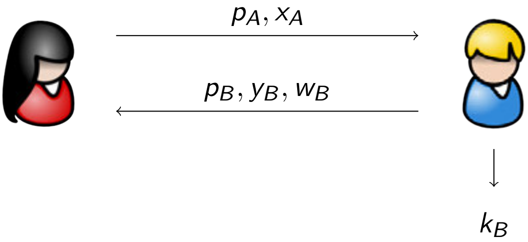
# HMQV from Ideal Lattices—Corrected

$$\xrightarrow{\quad p_A, x_A \quad}$$

# HMQV from Ideal Lattices—Corrected



# HMQV from Ideal Lattices—Corrected



# HMQV from Ideal Lattices—Corrected

$$p_A, x_A \longrightarrow$$

$$\longleftarrow p_B, y_B, w_B$$

$$\downarrow \qquad\qquad \downarrow$$

$$k_A \qquad\qquad k_B$$

# HMQV from Ideal Lattices—Corrected

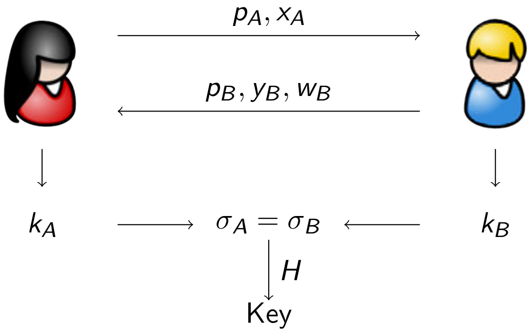
$$\xrightarrow{\quad p_A,\, x_A \quad}$$

$$\xleftarrow{\quad p_B,\, y_B,\, w_B \quad}$$

$$\downarrow \qquad\qquad\qquad\qquad \downarrow$$

$$k_A \longrightarrow \sigma_A = \sigma_B \longleftarrow k_B$$

# HMQV from Ideal Lattices—Corrected



# Proof Games

Proof proceeds by series of games:

- Begin with simulated protocol
- Replace one hash output with true random value, back-program random oracle
- Adversary cannot distinguish from previous game
- Eventually, if original protocol can be distinguished from random, rLWE can be broken
- The modification using rejecting sampling

# Forward Security

- If static keys compromised, previous session keys remain secure
- Notion captured in proof by giving adversaries ability to corrupt static key
- Use Bellare–Rogaway model restricted to two-pass

# Quantum Hardness

- Proof uses Random Oracle Model—quantum implications not fully understood
- Important step to post quantum key exchange

# Implementations Parameters

| Parameters | $n$ | Security (expt.) | $\alpha$ | $\gamma$ | $\log \frac{\beta}{\alpha}$ | $\log q$ (bits) |
|---|---|---|---|---|---|---|
| I* | 1024 | 80 bits | 3.397 | 101.919 | 8.5 | 40 |
| II | 2048 | 80 bits | 3.397 | 161.371 | 27 | 78 |
| III | 2048 | 128 bits | 3.397 | 161.371 | 19 | 63 |
| IV | 4096 | 128 bits | 3.397 | 256.495 | 50 | 125 |
| V | 4096 | 192 bits | 3.397 | 256.495 | 36 | 97 |
| VI | 4096 | 256 bits | 3.397 | 256.495 | 28 | 81 |

# Communication Overheads

| Choice of Parameters | Size (KB) | | | |
|---|---|---|---|---|
| | pk | sk (expt.) | init. msg | resp. msg |
| I* | 5 KB | 0.75 KB | 5 KB | 5.125 KB |
| II | 19.5 KB | 1.5 KB | 19.5 KB | 19.75 KB |
| III | 15.75 KB | 1.5 KB | 15.75 KB | 16 KB |
| IV | 62.5 KB | 3 KB | 62.5 KB | 63 KB |
| V | 48.5 KB | 3 KB | 48.5 KB | 49 KB |
| VI | 40.5 KB | 3 KB | 40.5 KB | 41 KB |

The bound $6\alpha$ with $\mathrm{erfc}(6) \approx 2^{-55}$ is used to estimate the size of secret keys.

# Timings

| Parameters | Initiation | Response | Finish |
|---|---|---|---|
| I | 3.22 ms (0.02 ms) | 8.50 ms (4.69 ms) | 5.23 ms (4.73 ms) |
| II | 12.00 ms (0.04 ms) | 29.33 ms (14.64 ms) | 17.28 ms (14.61 ms) |
| III | 10.33 ms (0.04 ms) | 25.83 ms (13.46 ms) | 15.58 ms (13.40 ms) |
| IV | 83.61 ms (0.08 ms) | 156.58 ms (39.86 ms) | 73.11 ms (39.73 ms) |
| V | 61.74 ms (0.08 ms) | 117.81 ms (32.58 ms) | 55.64 ms (32.20 ms) |
| VI | 25.42 ms (0.08 ms) | 62.31 ms (31.32 ms) | 36.80 ms (31.29 ms) |

Table: Timings of Proof-of-Concept Implementations in ms (The figures in the parentheses indicate the timings with pre-computing. For comparison, by simply using the "speed" command in openssl on the same machine, the timing for dsa1024 signing algorithm is about 0.7 ms, and for dsa2048 is about 2.3 ms).

## Summary

- We build a simple AKE based on RLWE.
- They are provably secure.
- We can prove the Forward Security of the AKE.
- Our preliminary implementations are very efficient.
  Our AKE are strong candidates for the post-quantum world.

# Work in Progress

- Password authenticated Key Exchange(PAKE) https://eprint.iacr.org/2016/552.pdf
- Authentication protocol using the signal functions.

# Work in Progress-Authentication Protocol

| Prover (P) | | Verifier (V) |
|---|---|---|
| Sample $s, e \leftarrow \chi_\alpha$<br>Secret Key: $s \in R_q$<br>Public Key: $p = as + e \in R_q$, $a$<br>Sample $s_1, e_1 \leftarrow \chi_\alpha$<br>Compute: $p_1 = as_1 + e_1 \in R_q$ | $\xrightarrow{p_1}$ | |
| | $\xleftarrow{x, b}$ | Sample $s', e' \leftarrow \chi_\alpha$<br>Set: $x = as' + e' \in R_q$<br>Random Challenge bit $b \leftarrow \{-1, 1\}$ |
| Sample $g_p \leftarrow \chi_\alpha$<br>Compute $k_p = (s_1 + bs)x + g_p$<br>$w = Sig(k_p)$ | $\xrightarrow{w}$ | Sample $g_v \leftarrow \chi_\alpha$<br>Compute $k_v = (p_1 + bp)s' + g_v$<br>Verify if $w$ match with the value of $k_v$ |

**Fig. 1.** Authenticated Protocol

# Thank You

Thank NIST and NSF for support!

# Thank you !

You can email your questions or comments to
jintai.ding@gmail.com