# Evaluating Post-Quantum Asymmetric Cryptographic Algorithm Candidates

Tolga Acar, Josh Benaloh, Craig Costello and **Dan Shumow**

MSR Security and Cryptography Group

Microsoft Research

# Motivation

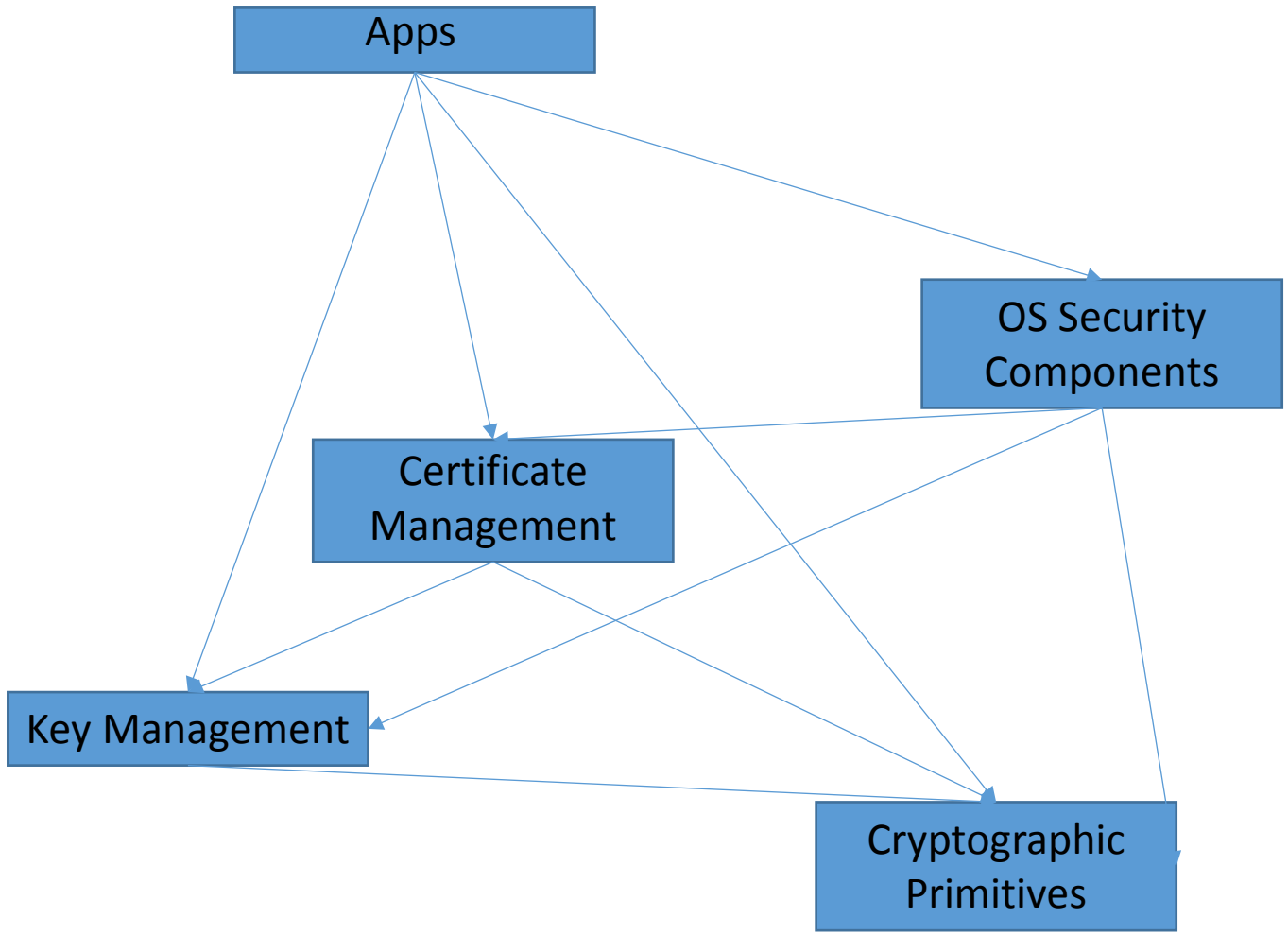
Obvious benefit of migrating to post-quantum crypto:
We are prepared for quantum computers **IF** they are ever made.

- Potentially small probability but very high payoff.

Other benefits:

- Cryptographic Agility
  - Software design principle to implement entire software stack so that it is easy to swap cryptographic primitives.
- Increase diversity of cryptosystems (reduce probability of related breaks.)

# Cryptographic Agility




Every level of this software stack:

- Can call into every layer below it. (And they do.)
- Should be coded in a way that abstracts the underlying algorithms including parameters such as key, signature or ciphtertext length. (And they don't.)

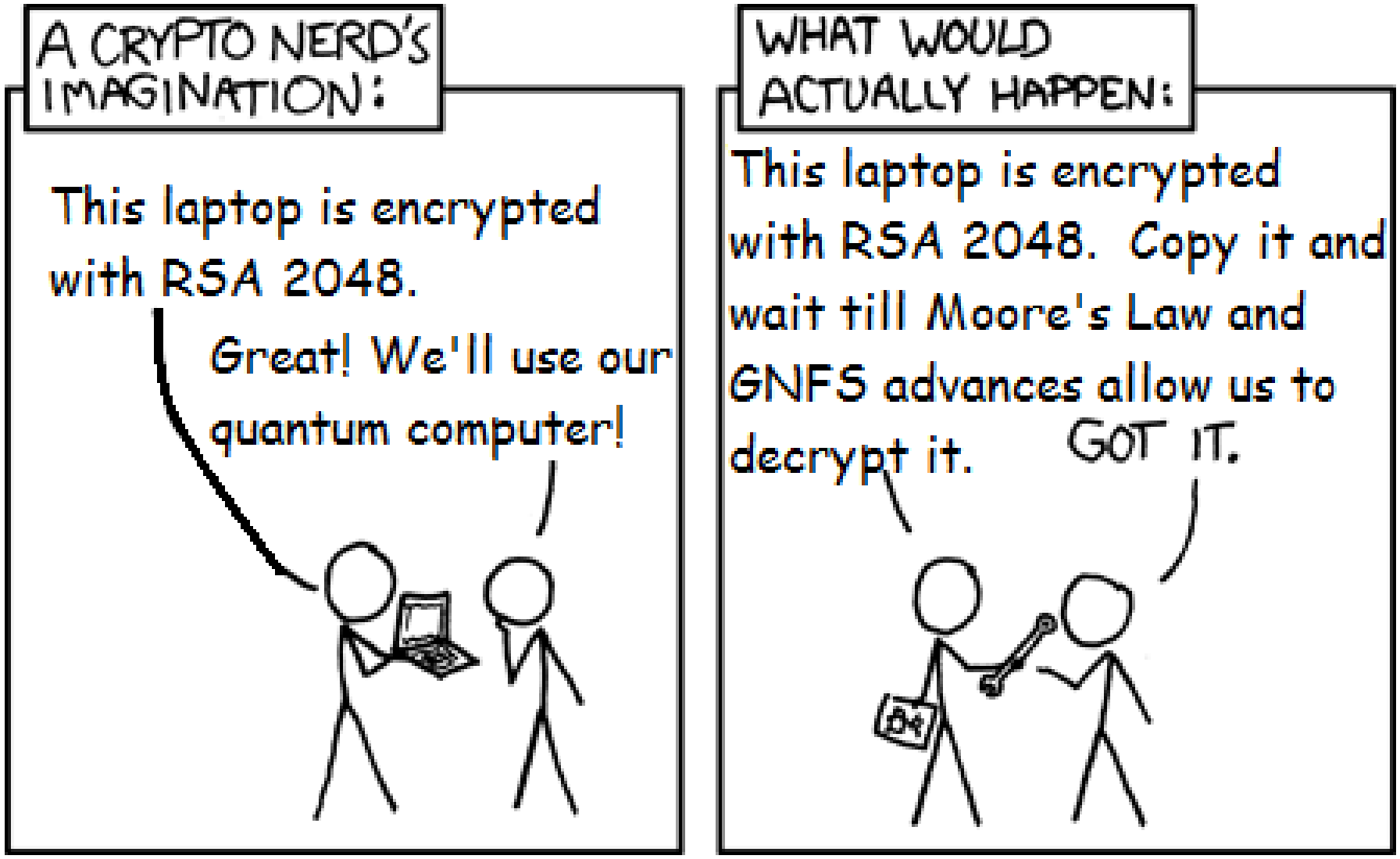
# Problems with Current Standards

Factoring may be hard in general -- But the margin between what can feasibly be factored and RSA key sizes is still vulnerable.

- Moore's law and advances in hardware erode security.
- State of the art in factoring algorithms has slowly but steadily progressed.

Current standardized asymmetric algorithms all lack diversity.

- Both factoring and discrete log attacks use the Number Field Sieve. Advances against RSA and DSA/DH may be related.
- ECC standardized groups only go up to the 256-bit security level. If algorithms for ECDLP advance, we have no higher security groups that we can use.




In actual reality: The key would have been generated with a broken RNG.

# Motivation (cont.)

Even if you don’t believe Quantum Computation will ever become a reality, there are benefits to implementing and deploying post quantum algorithms now.

Focus on signature and key exchange primitives.

Target scenarios for post-quantum algorithms:

- TLS
- Binary signatures
- Software update

CERTIFICATES CERTIFICATES CERTIFICATES

# Certificates

Currently:

- RSA Certificates: Low thousands of bytes.
- ECDSA Certificates: Hundreds of bytes.

Keeping TLS Certificates short is a major priority in networking:

- Major performance improvement:
  First message sent by the server contains the server certificate. Long certificates can cause message fragmentation, increasing latency and bandwidth usage. This can be more important than faster signing and verification.

# Belt and Suspenders: Use Two Signature

- Augment current signature algorithms with post quantum secure signature algorithms.
- When signing use both classical signature and a post-quantum signature.
- Verifiers only accept if both signatures are valid.

# Criteria for Evaluating Candidates

- Security: Both classical and quantum
- Ease of deployment:
  - Ideally we would like a drop in replacement for current algorithms.
  - Does the algorithm fit into existing abstractions?
  - Can we have manageably sized certificates?
    Key and signature length.
- Performance
  - Willing to take a hit as long as it is reasonable (≤10x increase.)

# Post-Quantum Signature Candidates

In this talk:

- Hash Based
- Code Based
- Permuted Kernel Problem Based
- LWE/RLWE Based

Not in this talk

- Super-Singular Isogeny Graph Based
- Multivariate Polynomial Based

# Hash Based Signatures

Evaluated Merkle variant of original Lamport scheme.

- Using private keys deterministically generated from a seed.

Parameters for the 128bit security level:

- Public Key: 256bits/32bytes
- Private Key: 256bits/32bytes
- Signature: 139,256bits/17,407bytes
- Lifetime: $2^{24}$ hashes

# Hash Based Signatures

Problems with this scheme:

- This algorithm does not fit the current abstraction of signing algorithms:
  - Managing keys that get “used up” over time is radically different than anything we have to do now.
- Signatures are very large.

Benefits of this scheme:

- The security is very well understood.

# Code Based Signatures

Using Niederreiter variant of McEliece.

For 128bit security level the following parameters

- Goppa codes, $n = 2960$, $k = 2288$ and $t = 56$
- Public Key 1,537,536bits/192,192bytes
- Private key 11,202,304bits/1,400,288bytes
- Signature 2,960bits/370bytes

All parameters taken from “Attacking and Defending McEliece” djb et al.

# Permuted Kernel Problem based Signatures

Permuted Kernel Problem (PKP):

Parameters: prime $q$ and positive integers $m$ and $n$ s.t. $m < n$.

1. Pick a random matrix $A \in \boldsymbol{Z}_q^{m \times n}$ and random permutation $\pi$ that permutes $n$ element vectors.
2. Find $n$ element vector $u \in \ker(A)$ and let $v = \pi(u)$.

Problem: Find $u$ given $A$ and $v$.

# Permuted Kernel Problem based Signatures

This is NP complete

- Reduction from the modular partition problem: Partition a set $S$ of $n$ integers, modulo a prime $q$ into two sets that sum to the same value modulo $q$.
- Let $A$ by a $1 \times n$ matrix with elements equal to those of set $S$. Let $v$ be a $n \times 1$ vector consisting of elements $1$ or $-1$. This is an instance of the PKP with solution if and only if $S$ has a partition.
- Vary the number of elements $-1$ in $v$ from 1 to $n - 1$.
- Use the PKP to build an identification scheme and use standard techniques to transform the identification scheme into a digital signature scheme.

# Permuted Kernel Problem based Signatures

This is NP complete

- Reduction from the modular partition problem: Partition a set $S$ of $n$ integers, modulo a prime $q$ into two sets that sum to the same value modulo $q$.
- Let $A$ by a $1 \times n$ matrix with elements equal to those of set $S$. Let $v$ be a $n \times 1$ vector consisting of elements $1$ or $-1$. This is an instance of the PKP with solution if and only if $S$ has a partition.
- Vary the number of elements $-1$ in $v$ from 1 to $n - 1$.

# Permuted Kernel Problem based Signatures

Use the PKP to build an identification scheme and use standard techniques to transform the identification scheme into a digital signature scheme.

Parameters for the 128bit security level:

- $q$ should be an 8bit prime (251), $m = 37$ and $n = 64$.
- Public Key: 512bits/64bytes
- Private Key: 512bits/64bytes
- Signature: 131,072 bits/16,384bytes

# Permuted Kernel Problem based Signatures

Security Considerations:

- This problem is closely related to knapsack problems, which are well known to have failed as the basis of cryptosystems.
- Otherwise stated: we don’t have a good idea of the average case hardness of this problem.

Benefits of PKP:

- Small prime modulus allows fast arithmetic.
- Fits into current abstractions of signature algorithms.

# LWE/RLWE Based Signatures

Based on success of RLWE Key Exchange this is our most promising candidate.

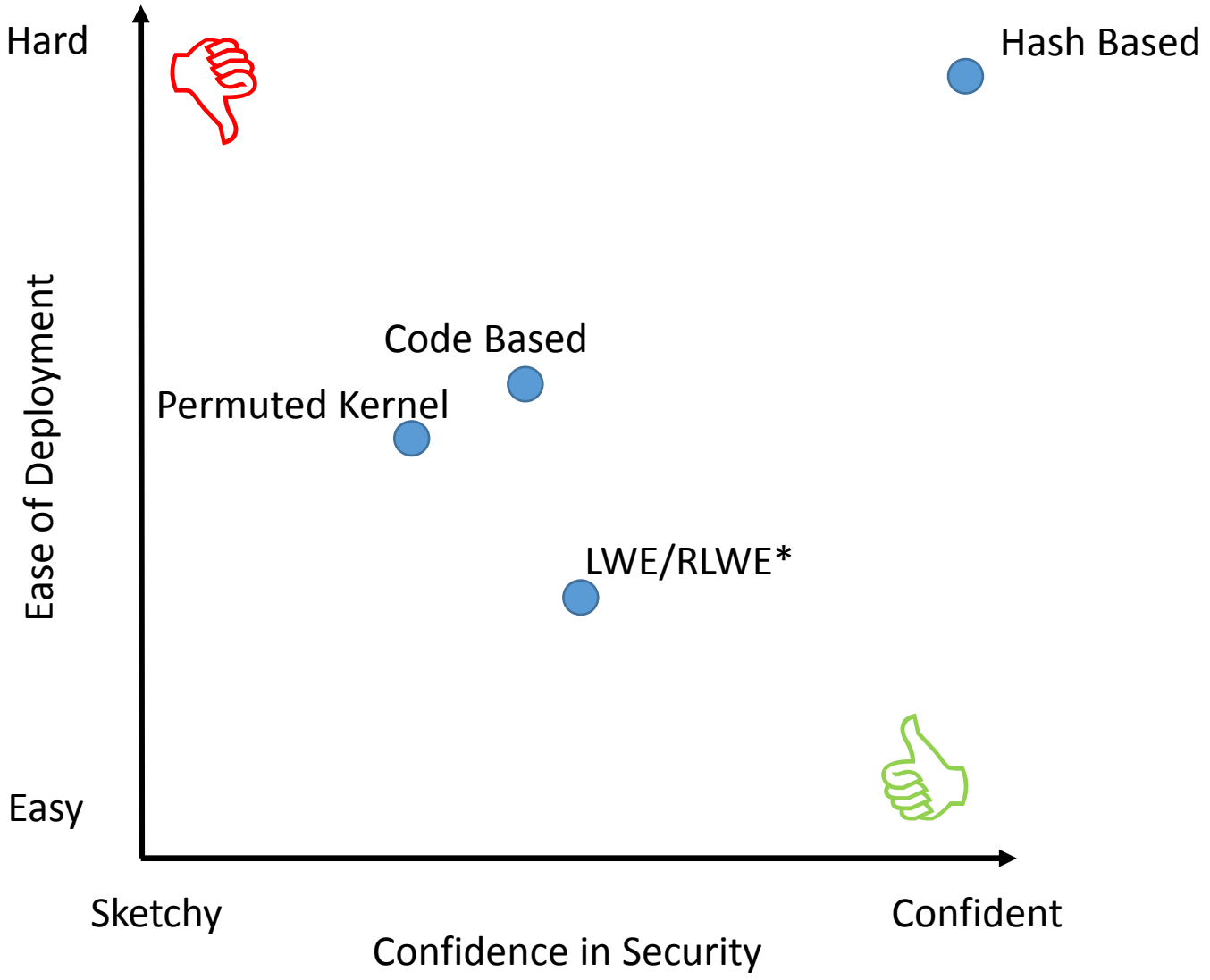
- We have only begun evaluating LWE/RLWE based signature schemes.
- Assuming parameter sizes for public, private and signature sizes are approximately the same (< 10kb).

# Parameter Size Comparison

| | Hash Based | Code Based | PKP Based |
|---|---|---|---|
| Public Key Size (bytes) | 64 | 192,192 | 32 |
| Private Key Size (bytes) | 64 | 1,400,288 | 32 |
| Signature Size (bytes) | 16384 | 370 | 17,407 |

128-bit security level

# Confidence-Difficulty Tradeoff



# Conclusion

There is no candidate that has everything we want.

Nothing works as well as RLWE Key Exchange.

All involve tradeoffs:

- Lamport hash based signature:
  - Great security.
  - Too difficult to use in practice: state management and large signatures.
- McEliece code based signature:
  - Large public keys.
  - Needs more security analysis, especially against Quantum attacks.
- Permuted Kernel based signature:
  - Large signatures.
  - Poorly understood security.

# Conclusion

Future Work:

- LWE/RLWE Signatures are the closest to meeting our criteria.*
  - Need more security analysis, especially for parameter selection.
  - Longer keys and signatures are relatively acceptable.
  - Performance improvements.
- Investigate super-singular elliptic curve isogeny graph based algorithms.

*If the parameter sizes are close to those for Key Exchange.

# Questions?