Cryptographic Assumptions

Advanced Topics in ~~Cybersecurity~~Cryptography (7CCSMATC)

Martin R. Albrecht

Outline

Digital Signatures

A signature scheme Σ consists of three PPT algorithms $(\mathsf{KeyGen}, \mathsf{Sign}, \mathsf{Verify})$ such that:

KeyGen The key generation algorithm is a randomised algorithm that takes as input a security parameter 1^λ and outputs a pair $(\mathsf{vk}, \mathsf{sk})$, the **verification key** and **signing key**, respectively.

- We write $(\mathsf{vk}, \mathsf{sk}) \leftarrow \mathsf{KeyGen}(1^\lambda)$.

Digital Signatures: Syntax and Correctness II

Sign The signing algorithm takes as input a signing key sk, a message m and outputs a signature σ. The signing algorithm may be randomised or deterministic.

- We write $\sigma \leftarrow \mathsf{Sign}(\mathsf{sk}, m)$.
- We may write $\sigma \leftarrow \mathsf{Sign}(\mathsf{sk}, m; r)$ to unearth the used randomness explicitly.

Digital Signatures: Syntax and Correctness III

Verify The verification algorithm takes as input a verification key vk, a signature σ and a message m and outputs a bit b, with $b = 1$ meaning the signature is valid and $b = 0$ meaning the signature is invalid. Verify is a deterministic algorithm.

- We write $b \leftarrow \mathsf{Verify}(\mathsf{vk}, \sigma, m)$.

Digital Signatures: Syntax and Correctness IV

We require that except with negligible probability over $(\mathsf{vk}, \mathsf{sk}) \leftarrow \mathsf{KeyGen}(1^\lambda)$ and potentially the randomness used in Sign, that it holds:

$$\mathsf{Verify}(\mathsf{vk}, \mathsf{Sign}(\mathsf{sk}, m), m) = 1 \text{ for all } m.$$

Digital Signatures: Security ((E/S)UF-CMA, traditional)

"... unforgeability under chosen message attacks"

((E/S)UF-CMA

$\mathcal{Q} \leftarrow \emptyset;$

$\mathsf{vk}, \mathsf{sk} \leftarrow \Sigma.\mathsf{KeyGen}(1^\lambda);$

$(m^\star, \sigma^\star) \leftarrow \mathcal{A}^{\mathsf{S}}(\mathsf{vk});$

// EUF-CMA: "existential"

return $(m^\star, \cdot) \notin \mathcal{Q} \land \Sigma.\mathsf{Verify}(\mathsf{vk}, \sigma^\star, m^\star) = 1$

// SUF-CMA: "strong"

return $(m^\star, \sigma^\star) \notin \mathcal{Q} \land \Sigma.\mathsf{Verify}(\mathsf{vk}, \sigma^\star, m^\star) = 1$

$\mathsf{S}(m)$

$\sigma \leftarrow \Sigma.\mathsf{Sign}(\mathsf{sk}, m)$

$\mathcal{Q} \leftarrow \mathcal{Q} \cup \{(m, \sigma)\}$

return σ

$$\mathsf{Adv}^{\mathrm{(e/s)uf\text{-}cma}}_{\mathcal{A},\Sigma}(\lambda) := \Pr[\mathsf{(e/s)uf\text{-}cma}^{\mathcal{A}}_{\Sigma}(\lambda) \Rightarrow 1]$$

Indistinguishability-based Definition

SUF-CMA

$\mathcal{Q} \leftarrow \emptyset;$
$b \leftarrow \{0,1\}$
$\mathsf{vk}, \mathsf{sk} \leftarrow \Sigma.\mathsf{KeyGen}(1^\lambda);$
$b' \leftarrow \mathcal{A}^{\mathsf{S},\mathsf{V}}(\mathsf{vk});$
return $b = b'$

$\mathsf{S}(m)$

$\sigma \leftarrow \Sigma.\mathsf{Sign}(\mathsf{sk}, m)$
$\mathcal{Q} \leftarrow \mathcal{Q} \cup \{(m, \sigma)\}$
return σ

$\mathsf{V}(m, \sigma)$

if $b = 1$ **then**
  return $\Sigma.\mathsf{Verify}(\mathsf{vk}, m, \sigma)$
else
  return $(m, \sigma) \in \mathcal{Q}$

Mike Rosulek. The Joy of Cryptography. https://joyofcryptography.com. self published, 2021

Cryptographic Assumptions

Sandcastles?

'Cryptographers seldom sleep well' (Silvio Micali). Their careers are frequently based on very precise complexity-theoretic assumptions, which could be shattered the next morning. A polynomial time algorithm for factoring would certainly prove more crushing than any paltry fluctuation of the Dow Jones." — Joe Kilian. Founding Cryptography on Oblivious Transfer. In: *20th ACM STOC*. ACM Press, May 1988, pp. 20–31. DOI: `10.1145/62212.62215`

- **1997** Gödel Prize
- **2004** RSA Award for Excellence in Mathematics
- **2007** Member of the National Academy of Sciences
- **2007** Fellow of the IACR
- **2012** Turing Award in 2012
- **2017** ACM Fellow

Cryptographic Primitives

Symmetric Primitives

- Block and stream ciphers (AES, ChaCha20, ...)
- Authentication codes (HMAC, Poly1305, ...)
- Hash functions (SHA-2, SHA-3, ...)

Asymmetric Primitives

- Key agreement and public-key encryption (RSA, DH, ECDH, ...)
- Digital signatures (RSA, DSA, ECDSA, ...)

Minicrypt vs Cryptomania

Definition (OWF)

A function $f : \{0,1\}^* \to \{0,1\}^*$ is **one-way** if the following two conditions hold:

1. **(Easy to compute)** There exists a polynomial-time algorithm $\mathcal{A}_f(x)$ computing f, i.e. $\mathcal{A}_f(x) = f(x)$ for all x.
2. **(Hard to invert)** for every probabilistic polynomial-time $\mathcal{B}$, we have:

$$\Pr_{x \leftarrow \{0,1\}^n} \left[\mathcal{B}\left(1^n, f(x)\right) \in f^{-1}(f(x))\right] < 1/\text{poly}(n).$$

OWF_f

$$\begin{array}{ll} 1: & x \leftarrow\!\$\ \{0,1\}^n \\ 2: & x' \leftarrow \mathcal{A}^{f(x)} \\ 3: & \textbf{return } x' = x \end{array}$$

Assumption

We do not **know** if one-way functions exist!

One-Way Functions imply a lot

- OWFs $\Rightarrow$ PRGs[1]
- PRGs $\Rightarrow$ PRFs[2]
- PRFs $\Rightarrow$ PRPs[3]
- OWFs $\Rightarrow$ Digital Signatures (sketch this lecture)
- OWFs $\overset{?}{\Rightarrow}$ Key Exchange / Public Key Exchange (unlikely!)

[1] Manuel Blum and Silvio Micali. How to Generate Cryptographically Strong Sequences of Pseudorandom Bits. In: *SIAM Journal on Computing* 13.4 (1984), pp. 850–864.
[2] Oded Goldreich, Shafi Goldwasser, and Silvio Micali. How to Construct Random Functions. In: *Journal of the ACM* 33.4 (Oct. 1986), pp. 792–807. DOI: 10.1145/6490.6503.
[3] Michael Luby and Charles Rackoff. How to construct pseudorandom permutations from pseudorandom functions. In: *SIAM Journal on Computing* 17.2 (1988).

One-Time Digital Signatures from OWFs

- **KeyGen** $f(\cdot)$ is a one-way function. Sample random numbers

$$(a_{0,0}, a_{0,1}), (a_{1,0}, a_{1,1}), \ldots, (a_{|m|-1,0}, a_{|m|-1,1}).$$

  Publish $f(a_{i,j})$ for all $a_{i,j}$.
- **Sign** Let b_i be the bits of m. For each bit b_i, publish a_{i,b_i}.
- **Verify** Check that a_{i,b_i} indeed maps to $f(a_{i,j})$ in the public key.

Leslie Lamport. **Constructing Digital Signatures from a One-way Function.** Technical Report SRI-CSL-98. SRI International Computer Science Laboratory, Oct. 1979

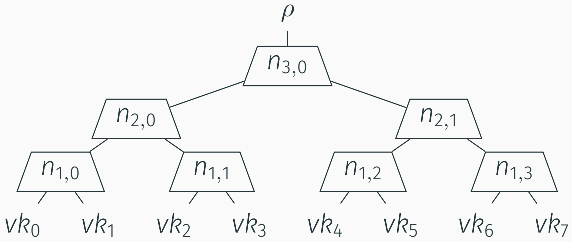
Toy Implementation

```python
from hashlib import sha256
from secrets import token_bytes as U
H = lambda m: sha256(m).digest()

def key_gen():
    sk = [(U(secpar//8), U(secpar//8)) for i in range(secpar)]
    vk = [(H(sk[i][0]), H(sk[i][1])) for i in range(secpar)]
    return vk, sk

def sign(sk, m):
    sig = []
    for i in range(secpar//8):
        for j in range(8):
            sig.append(sk[8*i+j][(m[i] >> j) % 2])
    return tuple(sig)

def verify(vk, sig, m):
    for i in range(secpar//8):
        for j in range(8):
            assert(vk[8*i+j][(m[i] >> j) % 2] ==  H(sig[8*i+j]))
    return True
```

```python
secpar = 256
vk, sk = key_gen()
m = U(secpar//8)
sig = sign(sk, m)
verify(vk, sig, m)
```

```
True
```

Merkle Trees



- publish $n_{3,0}$ as the vk
- pick a one-time signature scheme, e.g. vk_0 and sign with Lamport to produce σ.
- signature is $\sigma, vk_0, vk_1, n_{1,1}, n_{2,1}$

References

- Ralph Charles Merkle. **Secrecy, authentication, and public key systems.** PhD thesis. Stanford university, 1979
- Moni Naor and Moti Yung. **Universal One-Way Hash Functions and their Cryptographic Applications.** In: *21st ACM STOC*. ACM Press, May 1989, pp. 33–43. DOI: 10.1145/73007.73011 (based on OWFs)

“Blockchain”

Blockchains are degenerated Merkle trees.

Who would do such a thing?

Home | Resources | Software | Credits

SPHINCS+

SPHINCS+ is a stateless hash-based signature scheme, which was submitted to the NIST post-quantum crypto project. The design advances the SPHINCS signature scheme, which was presented at EUROCRYPT 2015. It incorporates multiple improvements, specifically aimed at reducing signature size. For a quick overview of the changes from SPHINCS to SPHINCS+ see the blog post by Andreas Hülsing. The submission proposes three different signature schemes:

- SPHINCS+-SHAKE256
- SPHINCS+-SHA-256
- SPHINCS+-Haraka

These signature schemes are obtained by instantiating the SPHINCS+ construction with SHAKE256, SHA-256, and

SPHINCS+ Team Leader and Primary Submitter

- Andreas Hülsing, Eindhoven University of Technology (NL)

SPHINCS+ Team

- Jean-Philippe Aumasson
- Daniel J. Bernstein, University of Illinois at Chicago (US) and Ruhr University Bochum (DE) and Academia Sinica (TW)
- Ward Beullens, KU Leuven (BE)
- Christoph Dobraunig, Graz University of Technology (AT)

Who would do such a thing?

NIST

Search CSRC

CSRC MENU

Information Technology Laboratory

COMPUTER SECURITY RESOURCE CENTER

PUBLICATIONS

FIPS 205 (Initial Public Draft)

Stateless Hash-Based Digital Signature Standard

Date Published: August 24, 2023
Comments Due: November 22, 2023 (public comment period is CLOSED)
Email Questions to: fips-205-comments@nist.gov

Author(s)

National Institute of Standards and Technology

Announcement

NIST requests comments on three draft Federal Information Processing Standards (FIPS):

- FIPS 203, *Module-Lattice-Based Key-Encapsulation Mechanism Standard*
- FIPS 204, *Module-Lattice-Based Digital Signature Standard*
- FIPS 205, *Stateless Hash-Based Digital Signature Standard*

These proposed standards specify key establishment and digital signature schemes that are designed to resist future attacks by quantum computers, which threaten the security of current standards. The three algorithms specified in these standards are each derived from different submissions to the NIST Post-Quantum

DOCUMENTATION

Publication:
https://doi.org/10.6028/NIST.FIPS.205.ipd
Download URL

Supplemental Material:
Federal Register Notice

Related NIST Publications:
FIPS 203 (Draft)
FIPS 204 (Draft)

Document History:
08/24/23: FIPS 205 (Draft)

TOPICS

Key Exchange from OWFs?

- Bob constructs n puzzles, each being an encryption of two random numbers x_i, y_i, and sends them to Alice.
- Alice picks a puzzle at random, say m_j, and brute-forces the decryption.[4]
- Alice encrypts her message using y_j as the key and sends x_j along in the clear.
- Bob uses x_j to identify which y_j to use for decryption.

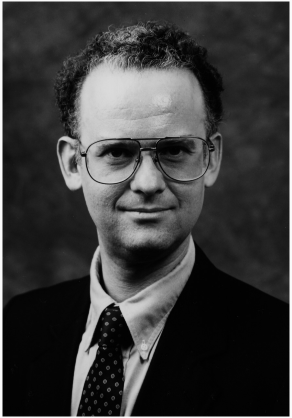
Ralph C Merkle. **Secure communications over insecure channels.** In: *Communications of the ACM* 21.4 (1978), pp. 294–299

[4] We assume this can be done in n steps and we assume Alice can decide when she guessed the correct decryption.

- Cost for Alice and Bob: $\mathcal{O}(n)$.
- Cost for an adversary: $\mathcal{O}(n^2)$: brute-force many puzzles to find the correct one

Boaz Barak and Mohammad Mahmoody-Ghidary. Merkle’s Key Agreement Protocol is Optimal: An $O(n^2)$ Attack on Any Key Agreement from Random Oracles. In: *Journal of Cryptology* 30.3 (July 2017), pp. 699–734. DOI: 10.1007/s00145-016-9233-9

Ralph Merkle



1996 Paris Kanellakis Award (from the ACM) for the Invention of Public Key Cryptography.

2000 RSA Award for Excellence in Mathematics for the invention of public key cryptography.

2008 International Association for Cryptographic Research (IACR) fellow for the invention of public key cryptography

2010 IEEE Hamming Medal for the invention of public key cryptography

2011 National Inventors Hall of Fame, for the invention of public key cryptography

Five Worlds

Algorithmica P = NP or something "morally equivalent" like fast probabilistic algorithms for NP.

Heuristica NP problems are hard in the worst case but easy on average.

Pessiland NP problems hard on average but no one-way functions exist. We can easily create hard NP problems, but not hard NP problems where we know the solution.

Minicrypt One-way functions exist but we do not have public-key cryptography.

Cryptomania Public-key cryptography is possible, i.e. two parties can exchange secret messages over open channels.

Credit: `https://blog.computationalcomplexity.org/2004/06/impagliazzos-five-worlds.html`

Russell Impagliazzo. A Personal View of Average-Case Complexity. In: *Proceedings of the Tenth Annual Structure in Complexity Theory Conference*. IEEE Computer Society, 1995, pp. 134–147. DOI: 10.1109/SCT.1995.514853. URL: https://doi.org/10.1109/SCT.1995.514853

Essential Cryptographic Primitives: Theoretical Perspective

Minicrypt

- Block and stream ciphers
- Hash functions[a]
- Authentication codes
- **Digital signatures**

[a] **Collision resistance** is not known to be in Minicrypt and evidence exists to the contrary: Nir Bitansky and Akshay Degwekar. On the Complexity of Collision Resistant Hash Functions: New and Old Black-Box Separations. In: *TCC 2019, Part I*. ed. by Dennis Hofheinz and Alon Rosen. Vol. 11891. LNCS. Springer, Cham, Dec. 2019, pp. 422–450. DOI: `10.1007/978-3-030-36030-6_17`

Cryptomania

- Key agreement and public-key encryption
- ...

Public-Key Encryption (PKE)

A Public-Key Encryption (PKE) scheme is a triple of PPT algorithms $(\mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec})$ with the following syntax and operation:

KeyGen The key generation algorithm is a randomised algorithm taking as input a security parameter 1^λ and outputs a public/secret key-pair $(\mathsf{pk}, \mathsf{sk})$, the **public key** and the **secret key** respectively.

Enc The encryption algorithm is a randomised algorithm taking as input a public-key pk and a message μ and outputs an encryption of m under pk.

Dec The decryption algorithm is a deterministic algorithm taking as input a ciphertext c and a secret-key sk, and outputs a message μ (or an error message indicating a decryption failure).

We require that except with negligible probability over $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{KeyGen}(1^\lambda)$ and and potentially the randomness used in Enc, that it holds:

$$\mathsf{Dec}(\mathsf{sk}, \mathsf{Enc}(\mathsf{pk}, \mu)) = \mu \text{ for all } \mu.$$

Key Encapsulation Mechanism (KEM)

A Key Encapsulation Mechanism (KEM) is a triple of PPT algorithms (KeyGen, Encap, Decap) with the following syntax and operation:

KeyGen The key generation algorithm is a randomised algorithm taking as input a security parameter 1^λ and outputs a public/secret key-pair $(\mathsf{pk}, \mathsf{sk})$, the **public key** and the **secret key** respectively.

Encap The encapsulation algorithm is a randomised algorithm taking as input a public-key pk and outputs a key $k \in \mathcal{K}$ and an encryption of k under pk.

Decap The decapsulation algorithm is a deterministic algorithm taking as input a ciphertext c and a secret-key sk, and outputs a key k (or an error message indicating a decryption failure).

We require that except with negligible probability over $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{KeyGen}(1^\lambda)$ and and potentially the randomness used in Encap, that it holds:

$$c, k \leftarrow\!\$\; \mathsf{Encap}(\mathsf{pk}), \mathsf{Dec}(\mathsf{sk}, c) = k.$$

IND-CPA of PKE

$\text{IND-CPA}_{\text{PKE}}$	$C(m_0, m_1)$
$\text{pk}, \text{sk} \leftarrow\!\!\$\ \text{KeyGen}(1^\lambda)$	**if** $\lvert m_0 \rvert \neq \lvert m_1 \rvert$ **then**
$b \leftarrow\!\!\$\ \{0,1\}$	**return** $\perp$
$b' \leftarrow \mathcal{D}^{C}$	$c \leftarrow\!\!\$\ \text{Enc}(\text{pk}, m_b)$
return $b = b'$	**return** c

Figure 1: IND-CPA Security Game (PKE).

$$\text{Adv}^{\text{ind-cpa}}_{PKE}(\mathcal{D}) = \lvert \Pr[\text{IND-CPA}^{\mathcal{D}} = 1] - 1/2 \rvert$$

IND-CPA of KEMs

$\text{IND-CPA}_{\text{KEM}}$	$C()$
$\mathsf{pk}, \mathsf{sk} \leftarrow\!\!\$\ \mathsf{KeyGen}(1^\lambda)$	$k_0 \leftarrow\!\!\$\ \mathcal{K}$
$b \leftarrow\!\!\$\ \{0, 1\}$	$c, k_1 \leftarrow\!\!\$\ \mathsf{Encap}(\mathsf{pk})$
$b' \leftarrow \mathcal{D}^{C}$	**return** c, k_b
return $b = b'$	

Figure 2: IND-CPA Security Game (KEM).

$$\mathsf{Adv}^{\text{ind-cpa}}_{KEM}(\mathcal{D}) = |\Pr[\text{IND-CPA}^{\mathcal{D}} = 1] - 1/2|$$

PKE <-> KEM

PKE $\rightarrow$ KEM Encrypt a random message.

KEM+Minicrypt $\rightarrow$ PKE Encrypt a key for a symmetric encryption scheme.

The Poverty of Public-Key Cryptography

Cryptographic Primitives

Symmetric Primitives

- Block and stream ciphers (AES, ChaCha20, …)
- Authentication codes (HMAC, Poly1305, …)
- Hash functions (SHA-2, SHA-3, …)

Asymmetric Primitives

- Key agreement and public-key encryption (RSA, DH, ECDH, …)
- Digital signatures (RSA, DSA, ECDSA, …)

The Wealth of Symmetric Cryptography

Indeed, it seems that "you can't throw a rock without hitting a one-way function" in the sense that, once you cobble together a large number of simple computational operations then, unless the operations satisfy some special property such as linearity, you will typically get a function that is hard to invert.[a]

[a] Boaz Barak. The Complexity of Public-Key Cryptography. Cryptology ePrint Archive, Report 2017/365. 2017. URL: `https://eprint.iacr.org/2017/365`.

The Wealth of Symmetric Cryptography?

Orr Dunkelman
@CryptoOrrDun

Unless you use Monkey Duplex, there is a lot of science involved in determining how many rounds are needed for security.

This tweet not only rude, it is ill-informed (and would help people believe they should write their own crypto, because if a monkey can, why can't they)

> **Boaz Barak** @boazbaraktcs · Oct 25, 2021
> Replying to @boazbaraktcs
> 2/20 Private key encryption often boils down to simply combining many non-local and non-linear operations. Making it efficient is challenging but if willing to lose some, a Monkey with a typewriter could probably construct a block cipher. See cambridge.org/core/journals/... by @wtgowers

12:25 PM · Oct 30, 2021

The Poverty of Public-Key Cryptography

The Internet runs on factoring and discrete logarithms

Rivest-Shamir-Adleman (RSA)

Factoring

Let p, q be primes of λ bits. Given $n := p \cdot q$, find p.

RSA

Let $n := p \cdot q$ be the product of two λ-bit primes. Let $e \nmid (p-1) \cdot (q-1)$. Given n, e and $c := p^e \bmod n$ for $p \leftarrow\!\!\$\ \mathbb{Z}_n$, find p.

Ronald L. Rivest, Adi Shamir, and Leonard M. Adleman. A Method for Obtaining Digital Signatures and Public-Key Cryptosystems. In: *Communications of the Association for Computing Machinery* 21.2 (Feb. 1978), pp. 120–126. DOI: 10.1145/359340.359342

- We'd like to say that RSA encryption/decryption is based on factoring, but such a reduction is not know
    - If factoring is easy then RSA is insecure
    - RSA could be insecure and factoring still hard[a]
- Rabin encryption ($\approx$ RSA with $e = 2$) is based on factoring

[a] … on a classical computer, we'll get to that shortly

Diffie-Hellman (DH)

Discrete Logarithms

Let p be a λ-bit prime and let g be a generator of the multiplicative group $\mathbb{Z}_p^*$. Given $g^a \bmod p$ find a.

DH

Let p be a λ-bit prime and let g be a generator of the multiplicative group $\mathbb{Z}_p^*$. Given $g^a \bmod p$, $g^b \bmod p$ and u, decide if $u = g^{ab}$ or random.

Whitfield Diffie and Martin E. Hellman. New Directions in Cryptography. In: *IEEE Transactions on Information Theory* 22.6 (1976), pp. 644–654. DOI: 10.1109/TIT.1976.1055638

- We'd like to say that the DH key exchange is based on discrete logarithms, but such a reduction is not known
  - If discrete logs are easy, DH is insecure
  - DH could be easy and discrete logarithms could still be hard[a]

[a] … on a classical computer, we'll get to that shortly

- We didn't use any properties of $\mathbb{Z}_p^*$ except that it is a group where discrete logarithms are hard.
- Elliptic curves are also groups where it is believed to be hard to compute discrete logarithms[5]
  - Indeed, it is believed only generic algorithms apply, in contrast to $\mathbb{Z}_p^*$ $\Rightarrow$ much smaller parameters
- Elliptic curves are usually written additively and not multiplicative.

[5] on a classical computer ...

Diffie-Hellman (DH)

Discrete Logarithms

Let p be a λ-bit prime and let g be a generator of the multiplicative group $\mathbb{Z}_p^*$. Given $g^a \bmod p$ find a.

DH

Let p be a λ-bit prime and let g be a generator of the multiplicative group $\mathbb{Z}_p^*$. Given $g^a \bmod p$, $g^b \bmod p$ and u, decide if $u = g^{ab}$ or random.

Discrete Logarithms

Let $\mathcal{G}$ be a group of order p and let G be a generator of $\mathcal{G}$. Given $a \cdot G$ for $a \in \mathbb{Z}_p$ find a.[a]

DH

Let $\mathcal{G}$ be a group of order p and let G be a generator of $\mathcal{G}$. Given $(G, a \cdot G, b \cdot G, U)$ for $a, b \in \mathbb{Z}_p$ decide if $U = a \cdot b \cdot G$ or random in $\mathcal{G}$.

[a] Here, $a \cdot G$ means to add G to itself a times.

Post-Quantum Era

Quantum Computers

- A quantum computer makes use of quantum effects (superpositions and entanglement) to perform computations.
- Quantum computers are not **faster** than classical computers, they are **different**.
- Some computations are easy on a quantum computer that are – as far as we know – hard on a classical computer.

- Small universal quantum computers exist.
- Key challenge is to scale them up by making them more stable.
- There is a critical point where we can scale up further using error correction.

FINANCIAL TIMES

HOME WORLD UK COMPANIES TECH MARKETS GRAPHICS OPINION WORK & CAREERS LIFE & ARTS HOW TO SPEND IT

Latest on Quantum technologies

A quantum future could be bright if managed well

Rivals rubbish Google's claim of quantum supremacy

Quantum computing builds on Turing's legacy

Quantum technologies + Add to myFT

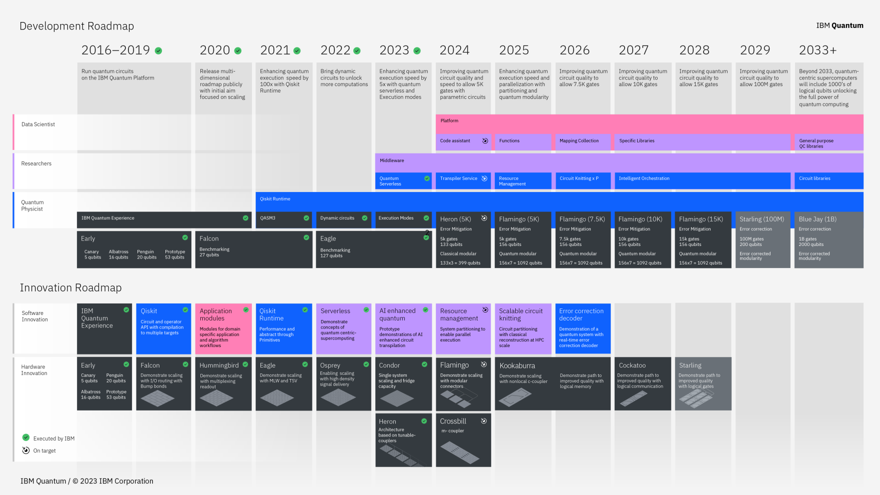
Google claims to have reached quantum supremacy

Researchers say their quantum computer has calculated an impossible problem for ordinary machines

Madhumita Murgia and Richard Waters SEPTEMBER 20 2019 162

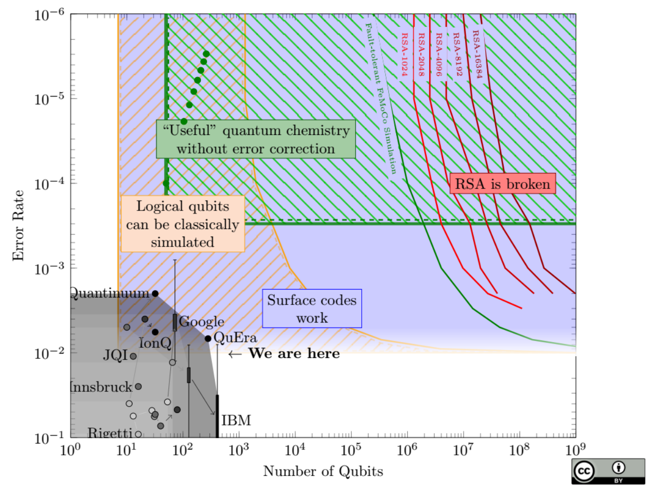
Google claims to have built the first quantum computer that can carry out calculations beyond the ability of today's most powerful supercomputers, a landmark moment that has been hotly anticipated by researchers.

IBM Quantum Computing Timeline



https://www.ibm.com/quantum/roadmap

Landscape of Quantum Computing in 2023



https://sam-jaques.appspot.com/quantum_landscape_2023

Symmetric Primitives: Quantum Computing Perspective (Good News)

Best known quantum algorithms for attacking symmetric cryptography are based on Grover's algorithm.

- Search key space of size 2^n in $2^{n/2}$ operations: AES-256 $\rightarrow$ 128 "quantum bits of security".
- Taking all costs into account: $> 2^{152}$ classical operations for AES-256.[6]
- Assuming a max depth of 2^{96} for a quantum circuit: overall AES-256 cost is $\approx 2^{190}$.
- Does not parallelise: have to wait for 2^X steps, cannot buy 2^{32} quantum computers and wait $2^X/2^{32}$ steps.

[6] Samuel Jaques, Michael Naehrig, Martin Roetteler, and Fernando Virdia. Implementing Grover Oracles for Quantum Key Search on AES and LowMC. In: *EUROCRYPT 2020, Part II*. ed. by Anne Canteaut and Yuval Ishai. Vol. 12106. LNCS. Springer, Cham, May 2020, pp. 280–310. DOI: `10.1007/978-3-030-45724-2_10`.

"BASED ON VERY PRECISE COMPLEXITY-THEORETIC ASSUMPTIONS, WHICH COULD BE SHATTERED THE NEXT MORNING"

Polynomial-Time Algorithms for Prime Factorization and Discrete Logarithms on a Quantum Computer*

Peter W. Shor†

Abstract

A digital computer is generally believed to be an efficient universal computing device; that is, it is believed able to simulate any physical computing device with an increase in computation time by at most a polynomial factor. This may not be true when quantum mechanics is taken into consideration. This paper considers factoring integers and finding discrete logarithms, two problems which are generally thought to be hard on a classical computer and which have been used as the basis of several proposed cryptosystems. Efficient randomized algorithms are given for these two problems on a hypothetical quantum computer. These algorithms take a number of steps polynomial in the input size, e.g., the number of digits of the integer to be factored.

"BASED ON VERY PRECISE COMPLEXITY-THEORETIC ASSUMPTIONS, WHICH COULD BE SHATTERED THE NEXT MORNING"

Post-Quantum Standardisation

- **NIST** **Post Quantum ~~Competition~~ Process**
- **ETSI** Cyber Working Group for Quantum Safe Cryptography
- **ISO** WG2 Standing Document 8 (SD8): Survey
- **IETF** Standardisation of **stateful** hash-based signatures, nothing further
- **CSA** Quantum-safe Security Working Group: position papers
- **NIST** **Post Quantum Process: Digital Signatures**

Post-Quantum Standardisation

NIST **Post Quantum ~~Competition~~ Process**

ETSI Cyber Working Group for Quantum Safe Cryptography

ISO WG2 Standing Document 8 (SD8): Survey

IETF Standardisation of **stateful** hash-based signatures, nothing further

CSA Quantum-safe Security Working Group: position papers

NIST **Post Quantum Process: Digital Signatures**

Bottom Line

Essentially, everyone is/was waiting for NIST.

NIST PQC ~~Competition~~ Process

Timeline

Submission	November 2017
Round 2 Selection	January 2019
Round 3 Selection	July 2020
Winners and Round 4 Selection	July 2022
3/4 Final Standards	August 2024
Final Standard for Falcon	???

"Key Establishment"/Key Encapsulation

- `(pk,sk) ← KeyGen()`
- `(c,k) ← Encap(pk)`
- `k ← Decap(c,sk)`

Digital Signature

- `(vk,sk) ← KeyGen()`
- `s ← Sig(m,sk)`
- `{0,1} ← Verify(s,m,vk)`

NIST PQC Outcome

NIST selected:

- **Kyber** A lattice-based KEM (MLWE Problem)
- **Dilithium** A lattice-based signature scheme (MSIS/MLWE Problems)
- **Falcon** A lattice-based signature scheme (NTRU Problem)
- **SPHINCS+** A hash-based signature scheme

"Based on very precise complexity-theoretic assumptions, which could be shattered the next morning" (SIKE Attack)

Wouter Castryck and Thomas Decru. An efficient key recovery attack on SIDH (preliminary version). Cryptology ePrint Archive, Report 2022/975. 2022. URL: https://eprint.iacr.org/2022/975

- SIDH was "A decade unscathed" [Cos21]
- SIKE even *lowered* parameters during NIST PQC (following [JS19])
- qualified researchers tried to break it (e.g. [MP19])

Total Break

All SIKE parameters can be broken in about 2 hours on a single-core laptop now [OP22].

Interpretation

This was an earthquake.

“Based on very precise complexity-theoretic assumptions, which could be shattered the next morning” (Rainbow Attack)

Breaking Rainbow Takes a Weekend on a Laptop

Ward Beullens

IBM Research, Zurich, Switzerland
wbe@zurich.ibm.com

Abstract. This work introduces new key recovery attacks against the Rainbow signature scheme, which is one of the three finalist signature schemes still in the NIST Post-Quantum Cryptography standardization project. The new attacks outperform previously known attacks for all the parameter sets submitted to NIST and make a key-recovery practical for the SL 1 parameters. Concretely, given a Rainbow public key for the SL 1 parameters of the second-round submission, our attack returns the corresponding secret key after on average 53 hours (one weekend) of computation time on a standard laptop.

- Rainbow was a NIST finalist[a]
- Can be remedied by increasing parameters, but makes it inefficient

[a] Ward Beullens. Breaking Rainbow Takes a Weekend on a Laptop. Cryptology ePrint Archive, Report 2022/214. 2022. URL: `https://eprint.iacr.org/2022/214`.

Interpretation

This was surprising.

NIST Signature Submissions with Vulnerabilities in Specification

3WISE,[7] AIMer, ALTEQ, Ascon-Sign, **Biscuit**,[8] CROSS, **DME-Sign**,[9] **EHT**,[10] **EagleSign**,[11] **Enhanced pqsigRM**,[12] FAEST, **FuLeeca**,[13] HAETAE, HAWK, **HPPC**,[14] **HuFu**,[15] **KAZ-SIGN**,[16] **LESS**,[17] MAYO, **MEDS**,[18] MIRA, MQOM, MiRitH, PERK, PROV, Preon, QR-UOV, RYDE, Raccoon, **SDitH**,[19] SNOVA, SPHINCS-alpha, SQIsign, SQUIRRELS, TUOV, UOV, VOX, Wave, **Xifrat1-Sign.I**,[20] eMLE-Sig 2.0

---

[7] https://groups.google.com/a/list.nist.gov/g/pqc-forum/c/fsfGqHCgGvY
[8] https://groups.google.com/a/list.nist.gov/g/pqc-forum/c/sw8NueiNek0
[9] https://groups.google.com/a/list.nist.gov/g/pqc-forum/c/E0mMMGI5eWE
[10] https://groups.google.com/a/list.nist.gov/g/pqc-forum/c/mFl_5Rq6-RU
[11] https://groups.google.com/a/list.nist.gov/g/pqc-forum/c/zas5PLiBe6A
[12] https://groups.google.com/a/list.nist.gov/g/pqc-forum/c/yQ1CKOLbGng
[13] https://groups.google.com/a/list.nist.gov/g/pqc-forum/c/KvIege2EbuM
[14] https://groups.google.com/a/list.nist.gov/g/pqc-forum/c/KRh8w03PW4E
[15] https://groups.google.com/a/list.nist.gov/g/pqc-forum/c/Hq-wRFDbIaU
[16] https://groups.google.com/a/list.nist.gov/g/pqc-forum/c/aCbi4BMDeUs
[17] https://groups.google.com/a/list.nist.gov/g/pqc-forum/c/Z36SPZJI8Ok
[18] https://groups.google.com/a/list.nist.gov/g/pqc-forum/c/CtCe8WXUoXI
[19] https://groups.google.com/a/list.nist.gov/g/pqc-forum/c/d_BcUfFGl5o
[20] https://groups.google.com/a/list.nist.gov/g/pqc-forum/c/9FXtBZKWueA

Interpretation

This is to be expected.

QKD?

"Given the specialised hardware requirements of QKD over classical cryptographic key agreement mechanisms and the requirement for authentication in all use cases, the NCSC does not endorse the use of QKD for any government or military applications, and cautions against sole reliance on QKD for business-critical networks, especially in Critical National Infrastructure sectors. [...] NCSC advice is that the best mitigation against the threat of quantum computers is quantum-safe cryptography."[21]

[21] https://www.ncsc.gov.uk/whitepaper/quantum-security-technologies

Thank You

References I

[Bar17] Boaz Barak. The Complexity of Public-Key Cryptography. Cryptology ePrint Archive, Report 2017/365. 2017. URL: `https://eprint.iacr.org/2017/365`.

[BD19] Nir Bitansky and Akshay Degwekar. On the Complexity of Collision Resistant Hash Functions: New and Old Black-Box Separations. In: *TCC 2019, Part I*. Ed. by Dennis Hofheinz and Alon Rosen. Vol. 11891. LNCS. Springer, Cham, Dec. 2019, pp. 422–450. DOI: `10.1007/978-3-030-36030-6_17`.

[Beu22] Ward Beullens. Breaking Rainbow Takes a Weekend on a Laptop. Cryptology ePrint Archive, Report 2022/214. 2022. URL: `https://eprint.iacr.org/2022/214`.

[BM17] Boaz Barak and Mohammad Mahmoody-Ghidary. Merkle's Key Agreement Protocol is Optimal: An $O(n^2)$ Attack on Any Key Agreement from Random Oracles. In: *Journal of Cryptology* 30.3 (July 2017), pp. 699–734. DOI: 10.1007/s00145-016-9233-9.

[BM84] Manuel Blum and Silvio Micali. How to Generate Cryptographically Strong Sequences of Pseudorandom Bits. In: *SIAM Journal on Computing* 13.4 (1984), pp. 850–864.

[CD22] Wouter Castryck and Thomas Decru. An efficient key recovery attack on SIDH (preliminary version). Cryptology ePrint Archive, Report 2022/975. 2022. URL: https://eprint.iacr.org/2022/975.

[Cos21] Craig Costello. The Case for SIKE: A Decade of the Supersingular Isogeny Problem. Cryptology ePrint Archive, Report 2021/543. 2021. URL: https://eprint.iacr.org/2021/543.

[DH76] Whitfield Diffie and Martin E. Hellman. New Directions in Cryptography. In: *IEEE Transactions on Information Theory* 22.6 (1976), pp. 644–654. DOI: `10.1109/TIT.1976.1055638`.

[GGM86] Oded Goldreich, Shafi Goldwasser, and Silvio Micali. How to Construct Random Functions. In: *Journal of the ACM* 33.4 (Oct. 1986), pp. 792–807. DOI: `10.1145/6490.6503`.

[Imp95] Russell Impagliazzo. A Personal View of Average-Case Complexity. In: *Proceedings of the Tenth Annual Structure in Complexity Theory Conference*. IEEE Computer Society, 1995, pp. 134–147. DOI: `10.1109/SCT.1995.514853`. URL: `https://doi.org/10.1109/SCT.1995.514853`.

[JNRV20] Samuel Jaques, Michael Naehrig, Martin Roetteler, and Fernando Virdia. **Implementing Grover Oracles for Quantum Key Search on AES and LowMC**. In: *EUROCRYPT 2020, Part II*. Ed. by Anne Canteaut and Yuval Ishai. Vol. 12106. LNCS. Springer, Cham, May 2020, pp. 280–310. DOI: 10.1007/978-3-030-45724-2_10.

[JS19] Samuel Jaques and John M. Schanck. **Quantum Cryptanalysis in the RAM Model: Claw-Finding Attacks on SIKE**. In: *CRYPTO 2019, Part I*. Ed. by Alexandra Boldyreva and Daniele Micciancio. Vol. 11692. LNCS. Springer, Cham, Aug. 2019, pp. 32–61. DOI: 10.1007/978-3-030-26948-7_2.

[Kil88] Joe Kilian. **Founding Cryptography on Oblivious Transfer**. In: *20th ACM STOC*. ACM Press, May 1988, pp. 20–31. DOI: 10.1145/62212.62215.

[Lam79] Leslie Lamport. **Constructing Digital Signatures from a One-way Function**. Technical Report SRI-CSL-98. SRI International Computer Science Laboratory, Oct. 1979.

[LR88] Michael Luby and Charles Rackoff. **How to construct pseudorandom permutations from pseudorandom functions.** In: *SIAM Journal on Computing* 17.2 (1988).

[Mer78] Ralph C Merkle. **Secure communications over insecure channels.** In: *Communications of the ACM* 21.4 (1978), pp. 294–299.

[Mer79] Ralph Charles Merkle. **Secrecy, authentication, and public key systems.** PhD thesis. Stanford university, 1979.

[MP19] Chloe Martindale and Lorenz Panny. **How to not break SIDH.** Cryptology ePrint Archive, Report 2019/558. 2019. URL: `https://eprint.iacr.org/2019/558`.

[NY89] Moni Naor and Moti Yung. **Universal One-Way Hash Functions and their Cryptographic Applications.** In: *21st ACM STOC*. ACM Press, May 1989, pp. 33–43. DOI: `10.1145/73007.73011`.

[OP22] Rémy Oudompheng and Giacomo Pope. A Note on Reimplementing the Castryck-Decru Attack and Lessons Learned for SageMath. Cryptology ePrint Archive, Report 2022/1283. 2022. URL: `https://eprint.iacr.org/2022/1283`.

[Ros21] Mike Rosulek. The Joy of Cryptography. `https://joyofcryptography.com`. self published, 2021.

[RSA78] Ronald L. Rivest, Adi Shamir, and Leonard M. Adleman. A Method for Obtaining Digital Signatures and Public-Key Cryptosystems. In: *Communications of the Association for Computing Machinery* 21.2 (Feb. 1978), pp. 120–126. DOI: `10.1145/359340.359342`.