Post-quantum cryptography

Joost Rijneveld

Digital Security,
Radboud University

2018-05-14
Colloquium Thalia

y tho

- Axiom: we want public-key cryptography
  - To exchange keys, to sign, . . . and do other things

y tho

- Axiom: we want public-key cryptography
  - To exchange keys, to sign, ... and do other things
- We have public-key cryptography
  - RSA, DH, ECC, ECDH, ...

y tho

- Axiom: we want public-key cryptography
  - To exchange keys, to sign, . . . and do other things

- We ~~have~~ had public-key cryptography
  - ~~RSA, DH, ECC, ECDH~~, . . .

Quantum computers!

y tho

- Axiom: we want public-key cryptography
  - To exchange keys, to sign, . . . and do other things

- We ~~have~~ had public-key cryptography
  - ~~RSA, DH, ECC, ECDH~~, . . .

Quantum computers!

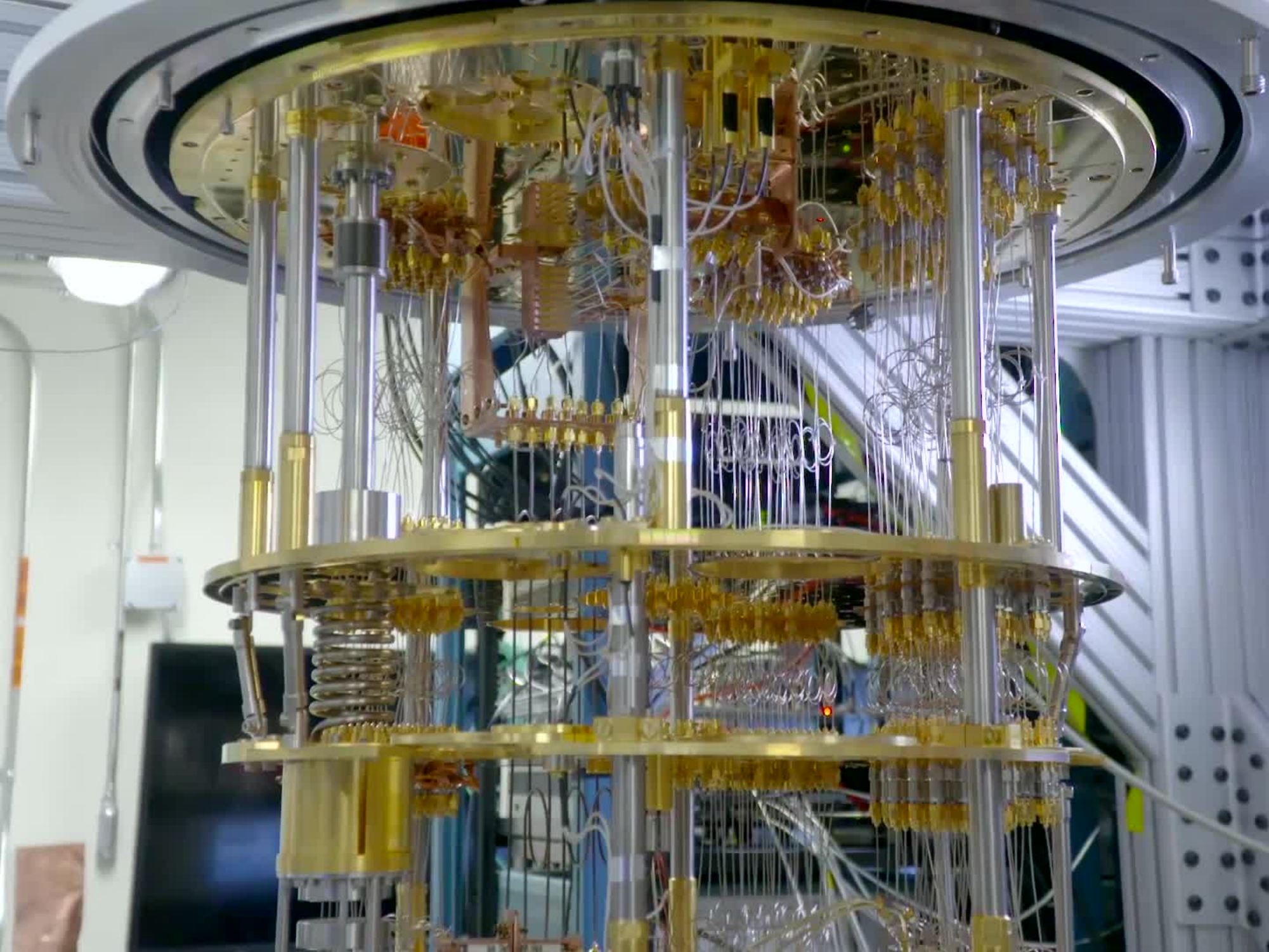
.. maybe

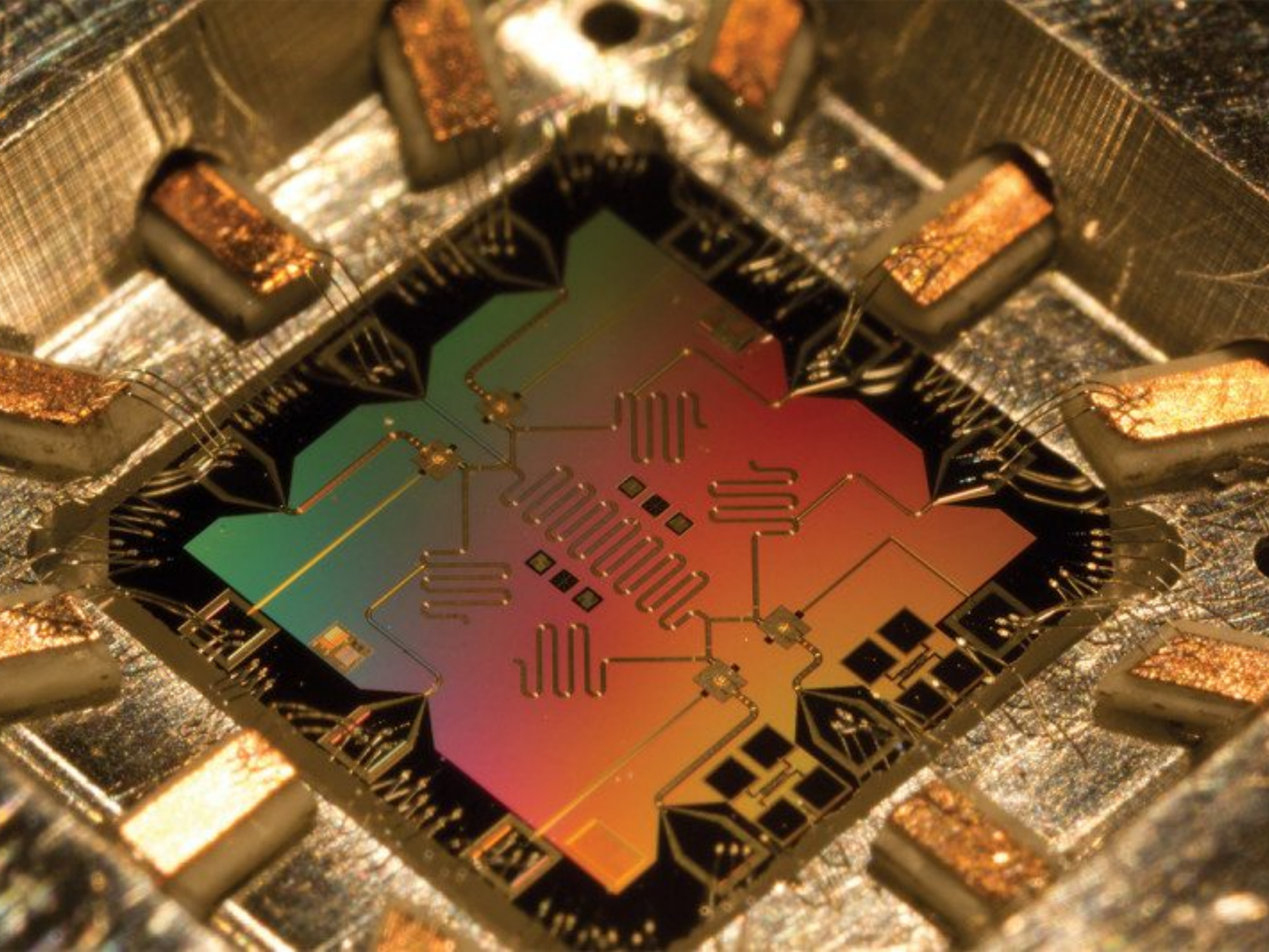
whoami

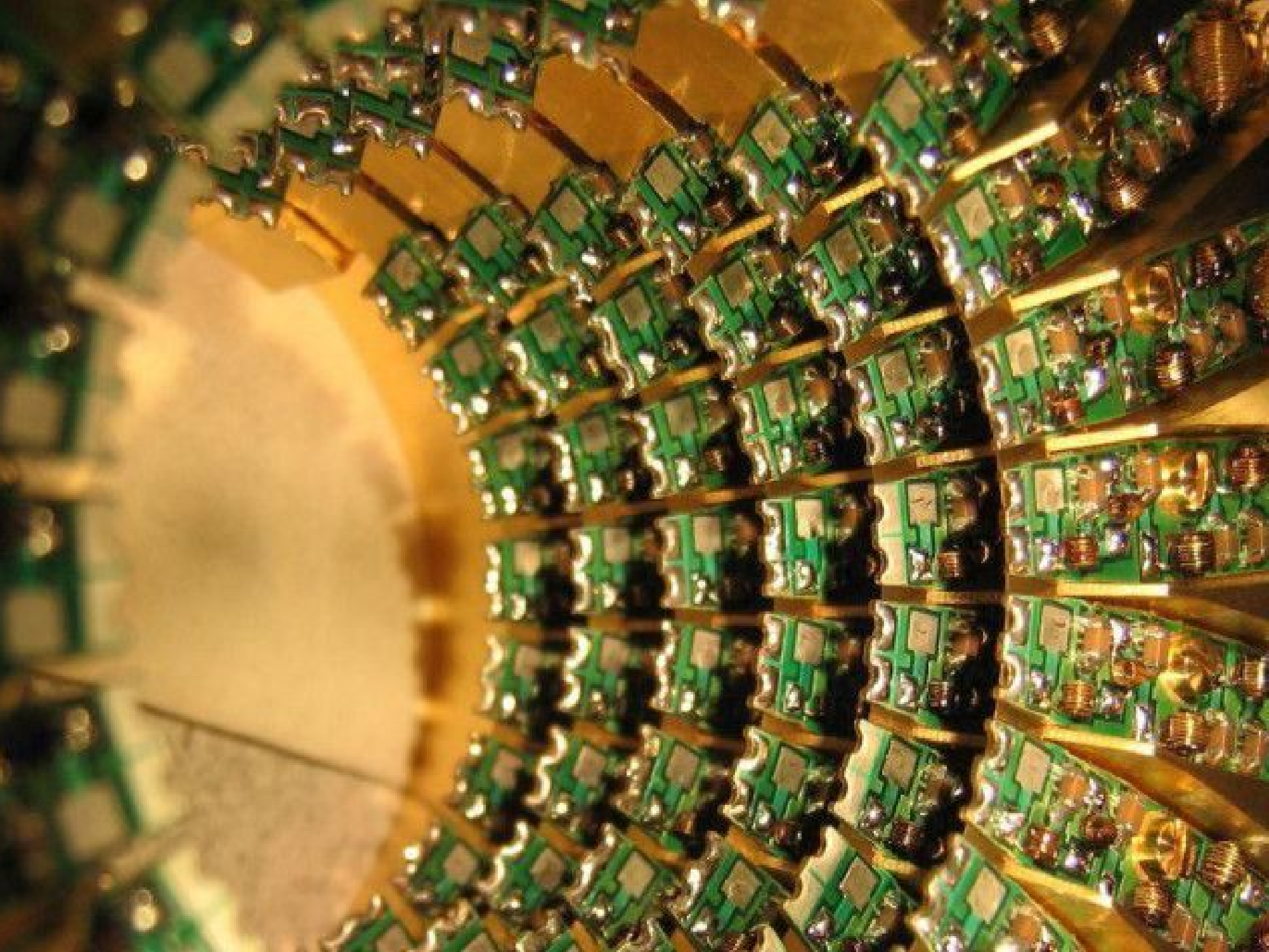
- PhD student at Digital Security
  - EU PQCRYPTO Project
  - Supervisor: Peter Schwabe
  - 'Cryptographic engineering'
    - Reference C, optimized assembly
    - Big Intels, small ARMs
- 2015 – 2019 (June?)

- 2013 – 2015 Kerckhoffs' Master (now TRU/e)
- 2010 – 2013 Computing Science Bachelor
  - Minor in Mathematics

What *is* a quantum computer?







D-Wave
2000Q
D-Wave
D-Wave
2000Q


D-Wave 2000Q
D-Wave 2000Q
D-Wave 2000Q

What is a quantum computer?

- .. I don't really know

What is a quantum computer?

- .. I don't really know
- But there's models

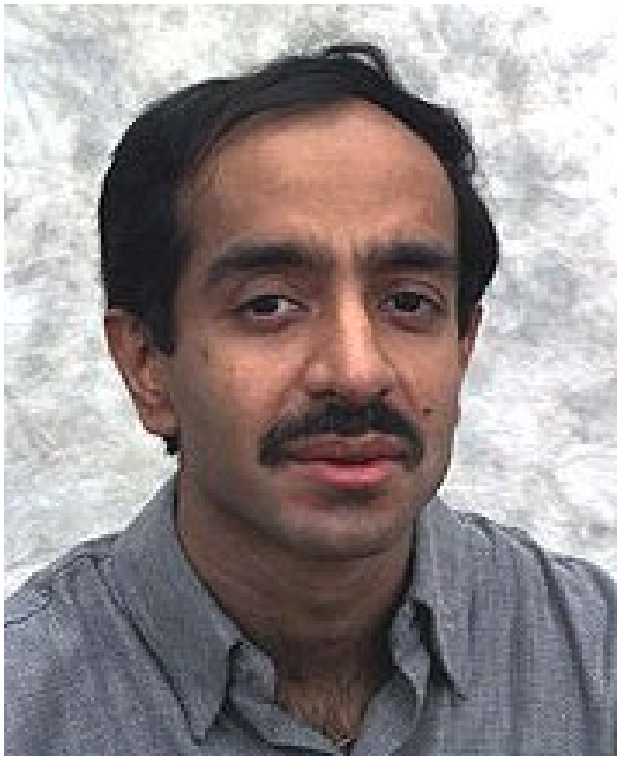
What is a quantum computer?

- .. I don't really know
- But there's models
- .. so I don't really care

What can it do?

- Useful things: complex simulations
  - Solve {global warming, world hunger, diseases, ...}
- Destructive things: break crypto

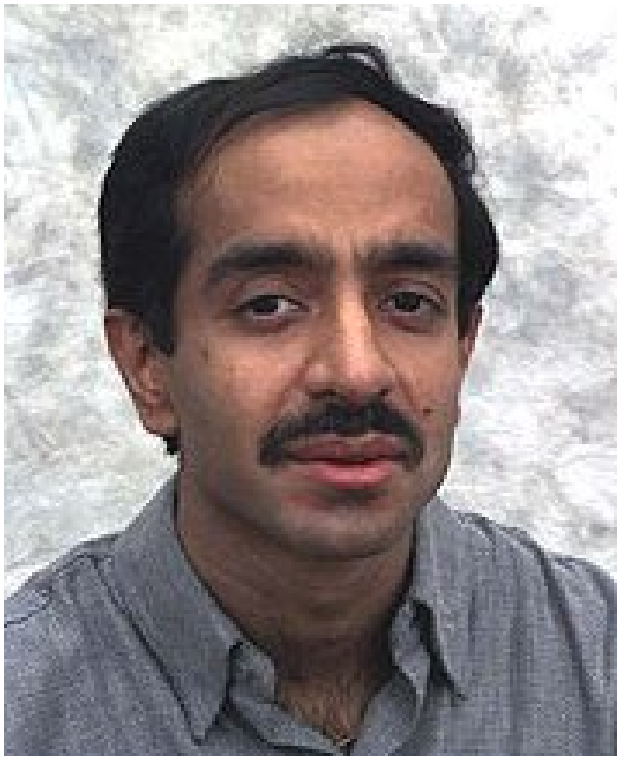
What can it do?



Grover: Search in $\mathcal{O}(\sqrt{n})$

Shor: Factorize in $poly(n)$

What can it do?



Grover: Search in $\mathcal{O}(\sqrt{n})$

Shor: Factorize in $poly(n)$

$\approx$ solve DLP

What can it do?

- Searching in $\mathcal{O}(\sqrt{n})$

- Factoring & solving DLP in $poly(n)$

What can it do?

- Searching in $\mathcal{O}(\sqrt{n})$
  - Brute force AES keys
  - Pre-image / collision search for hashes
- Factoring & solving DLP in $poly(n)$

What can it do?

- Searching in $\mathcal{O}(\sqrt{n})$
  - Brute force AES keys
  - Pre-image / collision search for hashes
  - Fix: double the lengths!
- Factoring & solving DLP in $poly(n)$

What can it do?

- Searching in $\mathcal{O}(\sqrt{n})$
  - Brute force AES keys
  - Pre-image / collision search for hashes
  - Fix: double the lengths!
- Factoring & solving DLP in $poly(n)$
  - Given $n = p \cdot q$, find p and q
  - Given $g^a \mod p$, find a

What can it do?

- Searching in $\mathcal{O}(\sqrt{n})$
  - Brute force AES keys
  - Pre-image / collision search for hashes
  - Fix: double the lengths!
- Factoring & solving DLP in $poly(n)$
  - Given $n = p \cdot q$, find p and q
  - Given $g^a \mod p$, find a
  - $poly(n)$ in asymptotics?

What can it do?

- Searching in $\mathcal{O}(\sqrt{n})$
  - Brute force AES keys
  - Pre-image / collision search for hashes
  - Fix: double the lengths!
- Factoring & solving DLP in $poly(n)$
  - Given $n = p \cdot q$, find p and q
  - Given $g^a \mod p$, find a
  - $poly(n)$ in asymptotics? Actually fast!

What can it do?

- Searching in $\mathcal{O}(\sqrt{n})$
  - Brute force AES keys
  - Pre-image / collision search for hashes
  - Fix: double the lengths!
- Factoring & solving DLP in $poly(n)$
  - Given $n = p \cdot q$, find p and q
  - Given $g^a \mod p$, find a
  - $poly(n)$ in asymptotics? Actually fast!
  - Fix: ..?

When though?

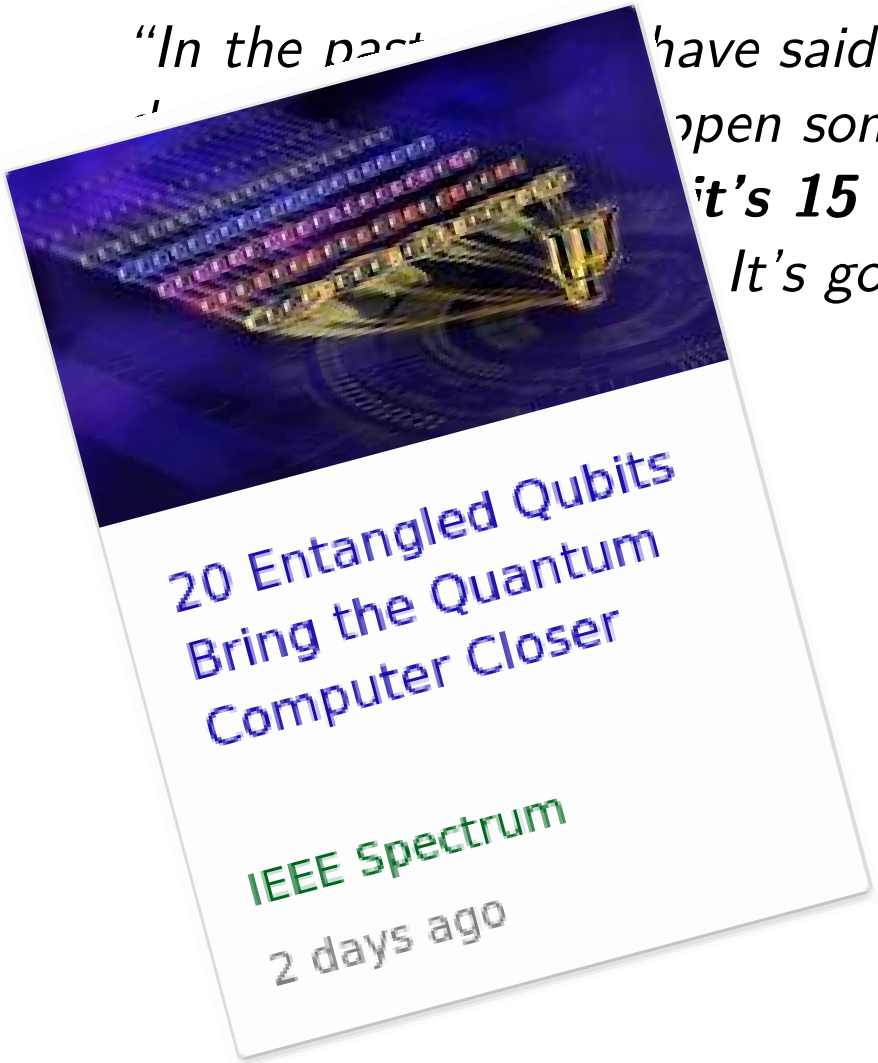
"In the past, people have said, maybe it's 50 years away, it's a dream, maybe it'll happen sometime. I used to think it was 50. Now I'm thinking like ***it's 15 or a little more****. It's within reach. It's within our lifetime. It's going to happen."*

— Mark Ketchen (IBM), Feb. 2012

When though?

*"In the past [illegible] have said, maybe it's 50 years away, it's a [illegible] open sometime. I used to think it was 50. **It's 15 or a little more**. It's within reach. It's going to happen."*

— Mark Ketchen (IBM), Feb. 2012



When though?

"In the past

ears away, it's a

think it was 50.

It's within reach.

en (IBM), Feb. 2012

20 Entangled
Bring the Qu
Computer C

IEEE Spectrum

2 days ago

Intelligent Machines

IBM Raises the Bar with a 50-Qubit Quantum Computer

Researchers have built the most sophisticated quantum computer yet, signaling progress toward a powerful new way of processing information.

by Will Knight November 10, 2017

IBM's 50-qubit machine.

When though?

"In the past

ears away, it's a

think it was 50.

It's within reach.

M), Feb. 2012

20 Entangled
Bring the Qu
Computer C

IEEE Spectrum

2 days ago

Intelligent Machines

IBM Raises the Bar with a 50-Qubit Quantum Computer

Rese
comp
way of

by Will K

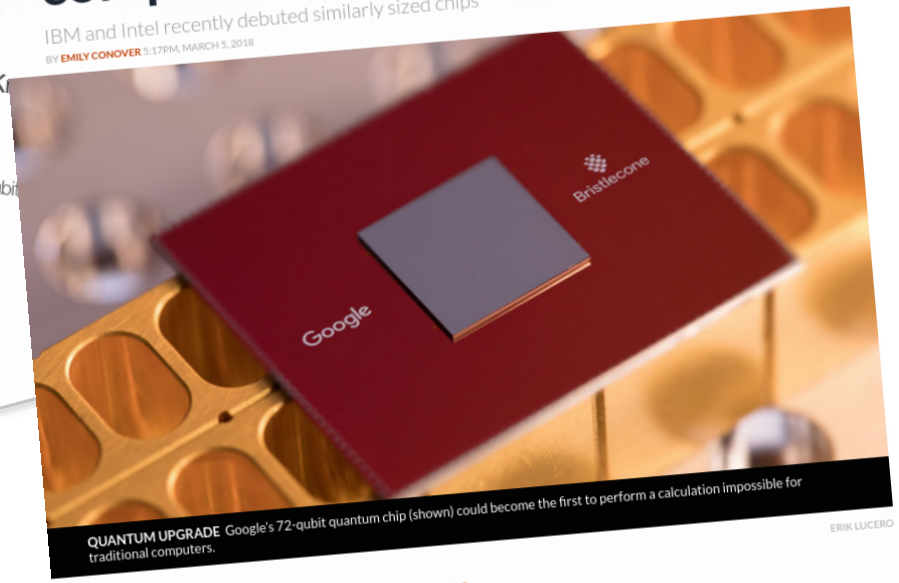
IBM's 50-qubi

NEWS IN BRIEF QUANTUM PHYSICS

Google moves toward quantum supremacy with 72-qubit computer

IBM and Intel recently debuted similarly sized chips

BY EMILY CONOVER 5:17PM, MARCH 5, 2018



QUANTUM UPGRADE Google's 72-qubit quantum chip (shown) could become the first to perform a calculation impossible for traditional computers.

ERIK LUCERO

Magazine issue: Vol. 193, No. 6, March 31, 2018, p. 13

When though?

"In the past

ears away, it's a

think it was 50.

It's within reach.

20 Entangled
Bring the Qu
Computer C

IEEE Spectrum

2 days ago

Intelligent Machines

IBM Raises the Bar with a 50-Qubit Quantum Com

Rese
comp
way of

by Will K

IBM's 50-qubi

NEWS IN BRIEF QUANTUM PHYSICS

Google moves supremacy w computer

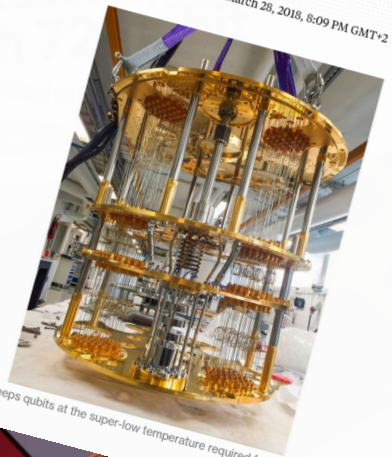
IBM and Intel recently debuted s

BY EMILY CONOVER 5:17PM, MARCH 5, 2018

Microsoft Edges Closer to Quantum Computer Based on Elusive Particle

Researchers make Majorana fermions, but now must try to control them

By Jeremy Kahn
March 28, 2018, 5:48 PM GMT+2 *Corrected* March 28, 2018, 8:09 PM GMT+2



A "quantum fridge" keeps qubits at the super-low temperature required for computing. *Source: Microsoft*

LIVE ON BLOOMBERG
Watch Live TV
Listen to Live Radio

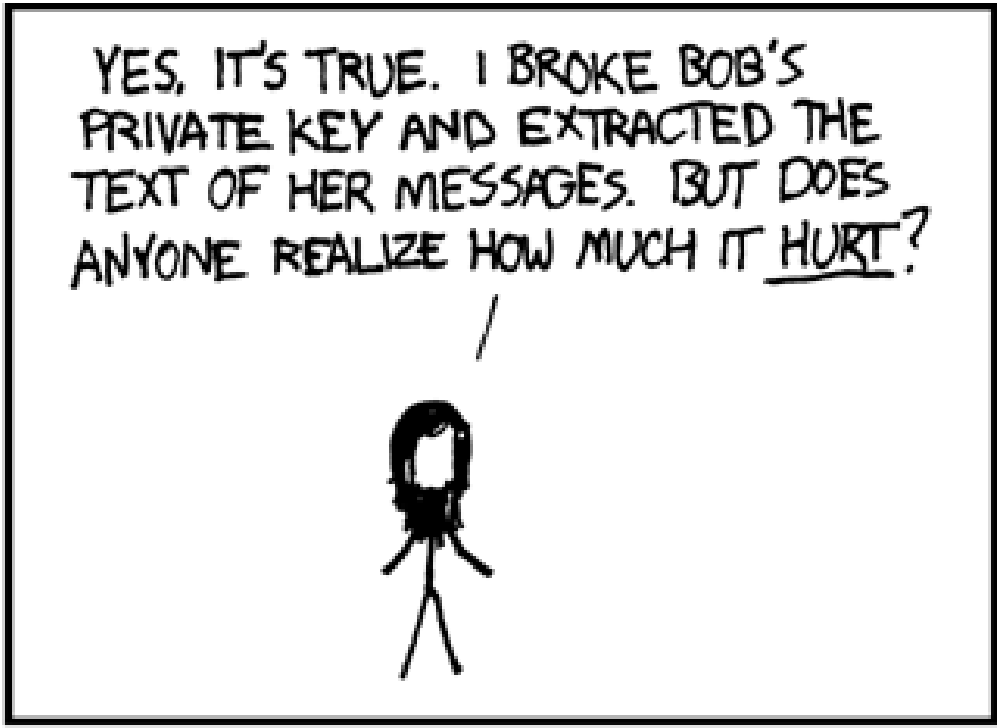
QUANTUM UPGRADE Google's 72-qubit quantum chip (shown) could become the first to perform a calculation impossible for traditional computers.

ERIK LUCERO

Magazine issue: Vol. 193, No. 6, March 31, 2018, p. 13

Attacker model

- Eve?



xkcd.com/177

- Or a **N**ation **S**tate **A**dversary?

See also: *'The Moral Character of Cryptographic Work'* by Phillip Rogaway

So all is lost?

- Symmetric crypto is fine!
  - Grover queries are expensive: AES-128 might be ‘ok’

So all is lost?

- Symmetric crypto is fine!
  - Grover queries are expensive: AES-128 might be 'ok'
- Asymmetric crypto is fun!
  - 99 problems, but the DLP ain't one

So all is lost?

- Symmetric crypto is fine!
  - Grover queries are expensive: AES-128 might be 'ok'
- Asymmetric crypto is fun!
  - 99 problems, but the DLP ain't one
  - Lattices $\mathbf{As} + \mathbf{e} \not\Rightarrow \mathbf{s}$

So all is lost?

- Symmetric crypto is fine!
  - Grover queries are expensive: AES-128 might be 'ok'
- Asymmetric crypto is fun!
  - 99 problems, but the DLP ain't one
  - Lattices $\mathbf{As} + \mathbf{e} \not\Rightarrow \mathbf{s}$
  - Error-correcting codes $\mathbf{m}\widehat{\mathbf{G}} + \mathbf{z} \not\Rightarrow \mathbf{m}$

So all is lost?

- Symmetric crypto is fine!
  - Grover queries are expensive: AES-128 might be 'ok'
- Asymmetric crypto is fun!
  - 99 problems, but the DLP ain't one
  - Lattices $\mathbf{As} + \mathbf{e} \not\Rightarrow \mathbf{s}$
  - Error-correcting codes $\mathbf{m}\widehat{\mathbf{G}} + \mathbf{z} \not\Rightarrow \mathbf{m}$
  - Multivariate quadratics $\mathbf{y} = \mathcal{MQ}(\mathbf{x})$

So all is lost?

- Symmetric crypto is fine!
  - Grover queries are expensive: AES-128 might be 'ok'
- Asymmetric crypto is fun!
  - 99 problems, but the DLP ain't one
  - Lattices $\mathbf{A}\mathbf{s} + \mathbf{e} \not\Rightarrow \mathbf{s}$
  - Error-correcting codes $\mathbf{m}\widehat{\mathbf{G}} + \mathbf{z} \not\Rightarrow \mathbf{m}$
  - Multivariate quadratics $\mathbf{y} = \mathcal{MQ}(\mathbf{x})$
  - Supersingular isogenies $\phi : E_1 \to E_2$

So all is lost?

- Symmetric crypto is fine!
  - Grover queries are expensive: AES-128 might be 'ok'
- Asymmetric crypto is fun!
  - 99 problems, but the DLP ain't one
  - Lattices $\mathbf{A}\mathbf{s} + \mathbf{e} \not\Rightarrow \mathbf{s}$
  - Error-correcting codes $\mathbf{m}\widehat{\mathbf{G}} + \mathbf{z} \not\Rightarrow \mathbf{m}$
  - Multivariate quadratics $\mathbf{y} = \mathcal{MQ}(\mathbf{x})$
  - Supersingular isogenies $\phi : E_1 \to E_2$
  - Hashes $\mathcal{H}(x) \not\Rightarrow x$
  - ...

So all is lost?

- Symmetric crypto is fine!
  - Grover queries are expensive: AES-128 might be 'ok'

- Asymmetric crypto is fun!
  - 99 problems, but the DLP ain't one

Lattices	$\mathbf{As} + \mathbf{e} \not\Rightarrow \mathbf{s}$
Error-correcting codes	$\mathbf{m}\widehat{\mathbf{G}} + \mathbf{z} \not\Rightarrow \mathbf{m}$
Multivariate quadratics	$\mathbf{y} = \mathcal{MQ}(\mathbf{x})$
Supersingular isogenies	$\phi : E_1 \to E_2$
Hashes	$\mathcal{H}(x) \not\Rightarrow x$
...	
post-quantum RSA	*'What if we used 1 GiB keys?'*

NIST Post-Quantum not-a-competition

- National Institute of Standards and Technology
  - See also: AES and SHA-3 competitions
- Deadline: November 30, 2017

NIST Post-Quantum not-a-competition

- National Institute of Standards and Technology
  - See also: AES and SHA-3 competitions
- Deadline: November 30, 2017
- 82 submissions

NIST Post-Quantum not-a-competition

- National Institute of Standards and Technology
  - See also: AES and SHA-3 competitions
- Deadline: November 30, 2017

- 82 submissions
- 69 ‘complete and proper’

NIST Post-Quantum not-a-competition

- National Institute of Standards and Technology
  - See also: AES and SHA-3 competitions
- Deadline: November 30, 2017

- 82 submissions
- 69 'complete and proper'
- $\approx$ 58 still unbroken

NIST Post-Quantum not-a-competition

- National Institute of Standards and Technology
  - See also: AES and SHA-3 competitions
- Deadline: November 30, 2017

- 82 submissions
- 69 'complete and proper'
- $\approx$ 58 still unbroken
- 8 with Radboud involved

NIST Post-Quantum not-a-competition

- National Institute of Standards and Technology
  - See also: AES and SHA-3 competitions
- Deadline: November 30, 2017

- 82 submissions
- 69 'complete and proper'
- $\approx$ 58 still unbroken
- 8 with Radboud involved

- PQC Standardization conference: April 11-13, 2018
- Final 'portfolio:' in 3 - 5 years

NIST Post-Quantum not-a-competition

- National Institute of Standards and Technology
  - See also: AES and SHA-3 competitions
- Deadline: November 30, 2017

- 82 submissions
- 69 'complete and proper'
- $\approx$ 58 still unbroken
- 8 with Radboud involved

- PQC Standardization conference: April 11-13, 2018
- Final 'portfolio:' in 3 - 5 years

- 'Not a competition'

Hash-based signatures

In a nutshell..

- Relies only on secure hash function
  - Pre-image resistance: $\mathcal{H}(x) \not\Rightarrow x$
  - No other assumptions
  - *The* conservative choice

In a nutshell..

- Relies only on secure hash function
  - Pre-image resistance: $\mathcal{H}(x) \not\Rightarrow x$
  - No other assumptions
  - *The* conservative choice

- Signatures are somewhat large ($\approx 8\,\mathrm{KiB}$)
- Signing is either slow or 'complicated'

In a nutshell..

- Relies only on secure hash function
  - Pre-image resistance: $\mathcal{H}(x) \not\Rightarrow x$
  - No other assumptions
  - *The* conservative choice

- Signatures are somewhat large ($\approx 8\,\text{KiB}$)
- Signing is either slow or 'complicated'

- Serious candidates for standardization
  - `draft-irtf-cfrg-xmss-hash-based-signatures`
  - SPHINCS$^+$ NIST submission
  - (Full disclosure: I'm involved in XMSS and SPHINCS$^+$)

In a nutshell..

- Relies only on secure hash function
  - Pre-image resistance: $\mathcal{H}(x) \not\Rightarrow x$
  - No other assumptions
  - *The* conservative choice

- Signatures are somewhat large ($\approx 8\,\text{KiB}$)
- Signing is either slow or 'complicated'

- Serious candidates for standardization RFC 8391
  - ~~`draft-irtf-cfrg-xmss-hash-based-signatures`~~
  - SPHINCS$^+$ NIST submission
  - (Full disclosure: I'm involved in XMSS and SPHINCS$^+$)

Authenticating a single bit

- Preparation step:
  - Generate s_{YES} and s_{NO} (large random values)

Authenticating a single bit

- Preparation step:
  - Generate s_{YES} and s_{NO} (large random values)
  - Publish $h(s_{YES})$ and $h(s_{NO})$

Authenticating a single bit

- Preparation step:
  - Generate s_{YES} and s_{NO} (large random values)
  - Publish $h(s_{YES})$ and $h(s_{NO})$

time passes

Authenticating a single bit

- Preparation step:
  - Generate s_{YES} and s_{NO} (large random values)
  - Publish $h(s_{YES})$ and $h(s_{NO})$

time passes

- Authentication step:
  - Publish s_{YES} or s_{NO} to authenticate 'YES' or 'NO'

Authenticating a single bit

- Preparation step:
  - Generate s_{YES} and s_{NO} (large random values)
  - Publish $h(s_{YES})$ and $h(s_{NO})$

time passes

- Authentication step:
  - Publish s_{YES} or s_{NO} to authenticate 'YES' or 'NO'

- Anyone can check and compare to hashes
- Can never re-use!

Lamport signatures

- ‘Classic example’ of hash-based signatures

Lamport signatures

- 'Classic example' of hash-based signatures
- Private key: N pairs of random numbers

$s_{0,0}$	$s_{1,0}$	$s_{2,0}$	$\cdots$	$s_{N-3,0}$	$s_{N-2,0}$	$s_{N-1,0}$
$s_{0,1}$	$s_{1,1}$	$s_{2,1}$	$\cdots$	$s_{N-3,1}$	$s_{N-2,1}$	$s_{N-1,1}$

Lamport signatures

- 'Classic example' of hash-based signatures
- Private key: N pairs of random numbers

$$\begin{array}{ccccccc} s_{0,0} & s_{1,0} & s_{2,0} & \cdots & s_{N-3,0} & s_{N-2,0} & s_{N-1,0} \\ s_{0,1} & s_{1,1} & s_{2,1} & \cdots & s_{N-3,1} & s_{N-2,1} & s_{N-1,1} \end{array}$$

- Public key: hashes of these random numbers

$$\begin{array}{ccccccc} \mathrm{h}(s_{0,0}) & \mathrm{h}(s_{1,0}) & \mathrm{h}(s_{2,0}) & \cdots & \mathrm{h}(s_{N-3,0}) & \mathrm{h}(s_{N-2,0}) & \mathrm{h}(s_{N-1,0}) \\ \mathrm{h}(s_{0,1}) & \mathrm{h}(s_{1,1}) & \mathrm{h}(s_{2,1}) & \cdots & \mathrm{h}(s_{N-3,1}) & \mathrm{h}(s_{N-2,1}) & \mathrm{h}(s_{N-1,1}) \end{array}$$

Lamport signatures

- 'Classic example' of hash-based signatures
- Private key: N pairs of random numbers

$$\begin{array}{ccccccc} s_{0,0} & s_{1,0} & s_{2,0} & \cdots & s_{N-3,0} & s_{N-2,0} & s_{N-1,0} \\ s_{0,1} & s_{1,1} & s_{2,1} & \cdots & s_{N-3,1} & s_{N-2,1} & s_{N-1,1} \end{array}$$

- Public key: hashes of these random numbers

$$\begin{array}{ccccccc} h(s_{0,0}) & h(s_{1,0}) & h(s_{2,0}) & \cdots & h(s_{N-3,0}) & h(s_{N-2,0}) & h(s_{N-1,0}) \\ h(s_{0,1}) & h(s_{1,1}) & h(s_{2,1}) & \cdots & h(s_{N-3,1}) & h(s_{N-2,1}) & h(s_{N-1,1}) \end{array}$$

- Signature on N-bit value, e.g. 100...110

$$s_{0,1} \quad s_{1,0} \quad s_{2,0} \quad \cdots \quad s_{N-3,1} \quad s_{N-2,1} \quad s_{N-1,0}$$

Lamport signatures

- 'Classic example' of hash-based signatures
- Private key: N pairs of random numbers

$$\begin{array}{ccccccc} s_{0,0} & s_{1,0} & s_{2,0} & \cdots & s_{N-3,0} & s_{N-2,0} & s_{N-1,0} \\ s_{0,1} & s_{1,1} & s_{2,1} & \cdots & s_{N-3,1} & s_{N-2,1} & s_{N-1,1} \end{array}$$

- Public key: hashes of these random numbers

$$\begin{array}{ccccccc} h(s_{0,0}) & h(s_{1,0}) & h(s_{2,0}) & \cdots & h(s_{N-3,0}) & h(s_{N-2,0}) & h(s_{N-1,0}) \\ h(s_{0,1}) & h(s_{1,1}) & h(s_{2,1}) & \cdots & h(s_{N-3,1}) & h(s_{N-2,1}) & h(s_{N-1,1}) \end{array}$$

- Signature on N-bit value, e.g. 100...110

$$s_{0,1} \quad s_{1,0} \quad s_{2,0} \quad \cdots \quad s_{N-3,1} \quad s_{N-2,1} \quad s_{N-1,0}$$

- Verification: hash, compare to public key

Lamport signatures

- 'Classic example' of hash-based signatures
- Private key: N pairs of random numbers

$$\begin{array}{ccccccc} s_{0,0} & s_{1,0} & s_{2,0} & \cdots & s_{N-3,0} & s_{N-2,0} & s_{N-1,0} \\ s_{0,1} & s_{1,1} & s_{2,1} & \cdots & s_{N-3,1} & s_{N-2,1} & s_{N-1,1} \end{array}$$

- Public key: hashes of these random numbers

$$\begin{array}{ccccccc} h(s_{0,0}) & h(s_{1,0}) & h(s_{2,0}) & \cdots & h(s_{N-3,0}) & h(s_{N-2,0}) & h(s_{N-1,0}) \\ h(s_{0,1}) & h(s_{1,1}) & h(s_{2,1}) & \cdots & h(s_{N-3,1}) & h(s_{N-2,1}) & h(s_{N-1,1}) \end{array}$$

- Signature on N-bit value, e.g. 100...110

$$\begin{array}{ccccccc} s_{0,1} & s_{1,0} & s_{2,0} & \cdots & s_{N-3,1} & s_{N-2,1} & s_{N-1,0} \end{array}$$

- Verification: hash, compare to public key
- Can still only do this **once**!

The Winternitz improvement

- Idea: sign groups of $\log(w)$ bits (let $w = 2^n$)
- Trade time for signature and key size

Note: 'checksum chains' to prevent forgery omitted for simplicity

The Winternitz improvement

- Idea: sign groups of $\log(w)$ bits (let $w = 2^n$)
- Trade time for signature and key size
- Example: $w = 4$, let's sign `10 00 11 01 01`

Note: 'checksum chains' to prevent forgery omitted for simplicity

The Winternitz improvement

- Idea: sign groups of $\log(w)$ bits (let $w = 2^n$)
- Trade time for signature and key size
- Example: $w = 4$, let's sign `10 00 11 01 01`

private key: s_0 s_1 s_2 s_3 s_4

Note: 'checksum chains' to prevent forgery omitted for simplicity

The Winternitz improvement

- Idea: sign groups of $\log(w)$ bits (let $w = 2^n$)
- Trade time for signature and key size
- Example: $w = 4$, let's sign `10 00 11 01 01`

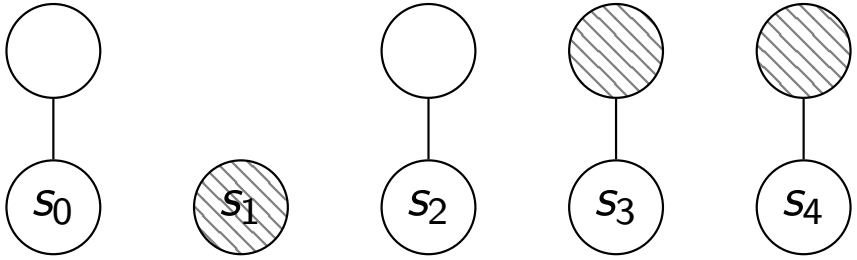
private key: s_0 s_1 s_2 s_3 s_4

Note: 'checksum chains' to prevent forgery omitted for simplicity

The Winternitz improvement

- Idea: sign groups of $\log(w)$ bits (let $w = 2^n$)
- Trade time for signature and key size
- Example: $w = 4$, let's sign `10 00 11 01 01`

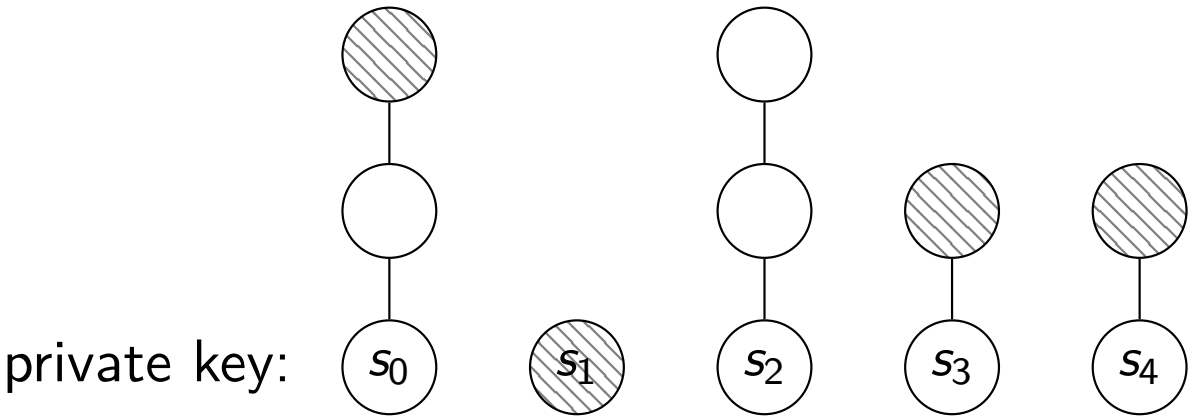
private key:



Note: 'checksum chains' to prevent forgery omitted for simplicity

The Winternitz improvement

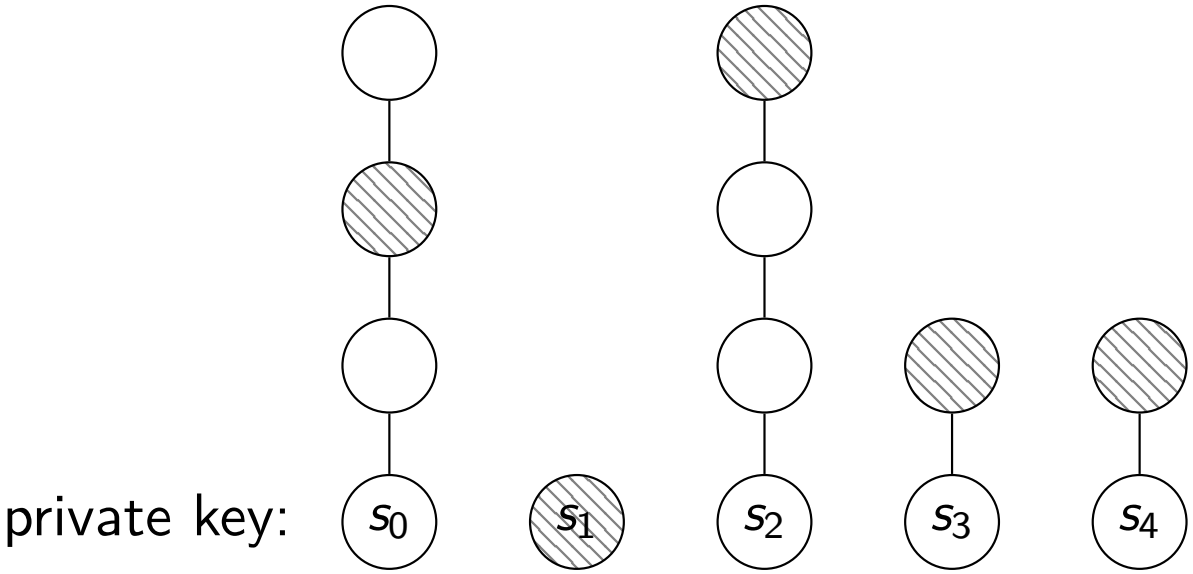
- Idea: sign groups of $\log(w)$ bits (let $w = 2^n$)
- Trade time for signature and key size
- Example: $w = 4$, let's sign `10 00 11 01 01`



Note: 'checksum chains' to prevent forgery omitted for simplicity

The Winternitz improvement

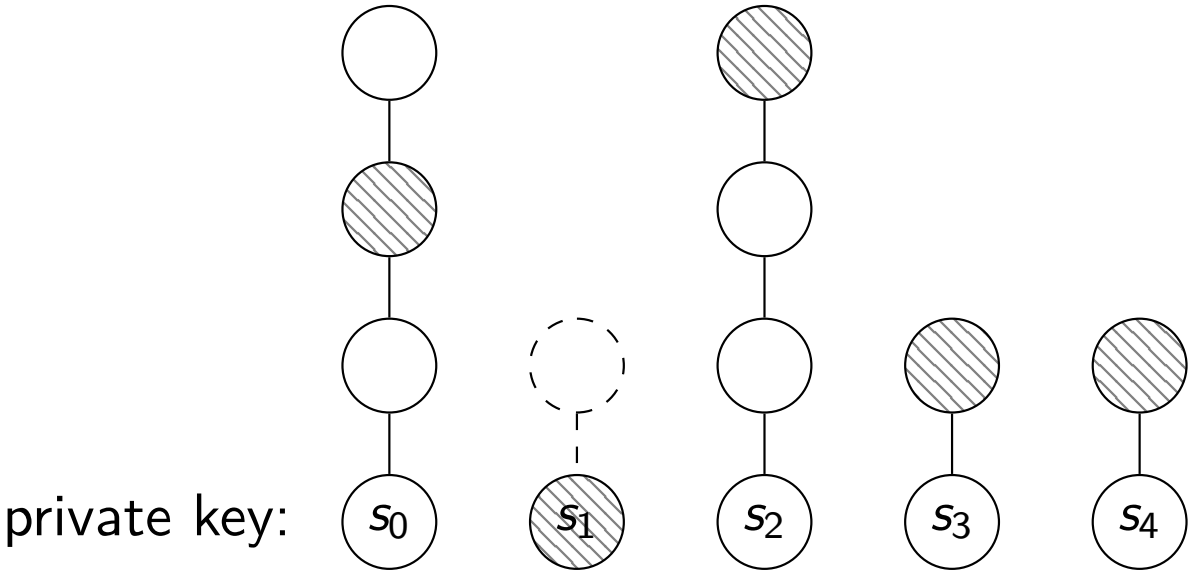
- Idea: sign groups of $\log(w)$ bits (let $w = 2^n$)
- Trade time for signature and key size
- Example: $w = 4$, let's sign `10 00 11 01 01`



Note: 'checksum chains' to prevent forgery omitted for simplicity

The Winternitz improvement

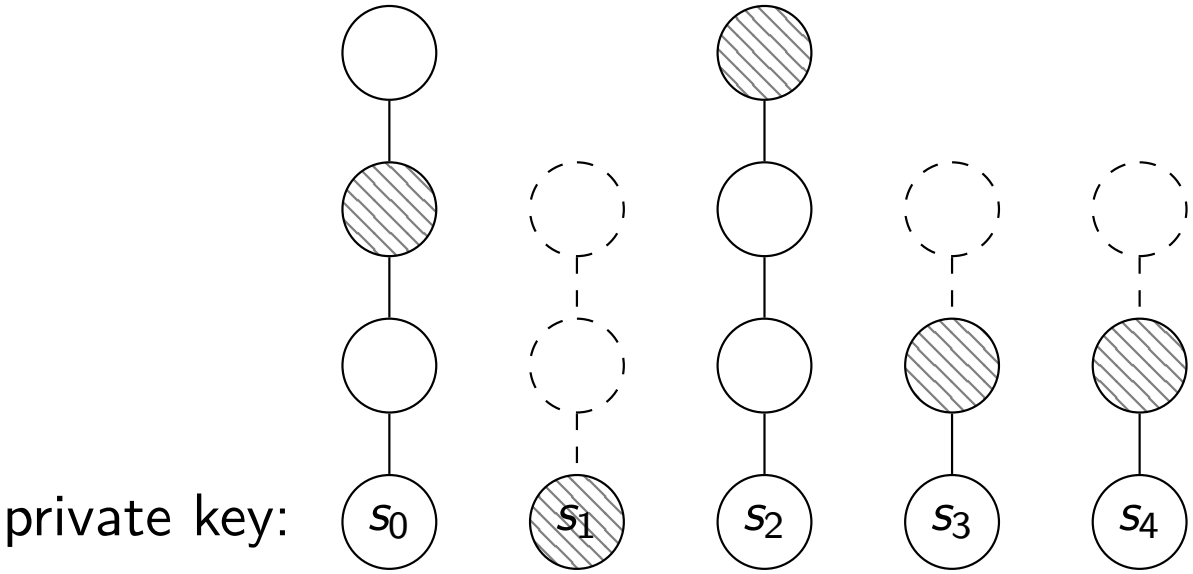
- Idea: sign groups of $\log(w)$ bits (let $w = 2^n$)
- Trade time for signature and key size
- Example: $w = 4$, let's sign `10 00 11 01 01`



Note: 'checksum chains' to prevent forgery omitted for simplicity

The Winternitz improvement

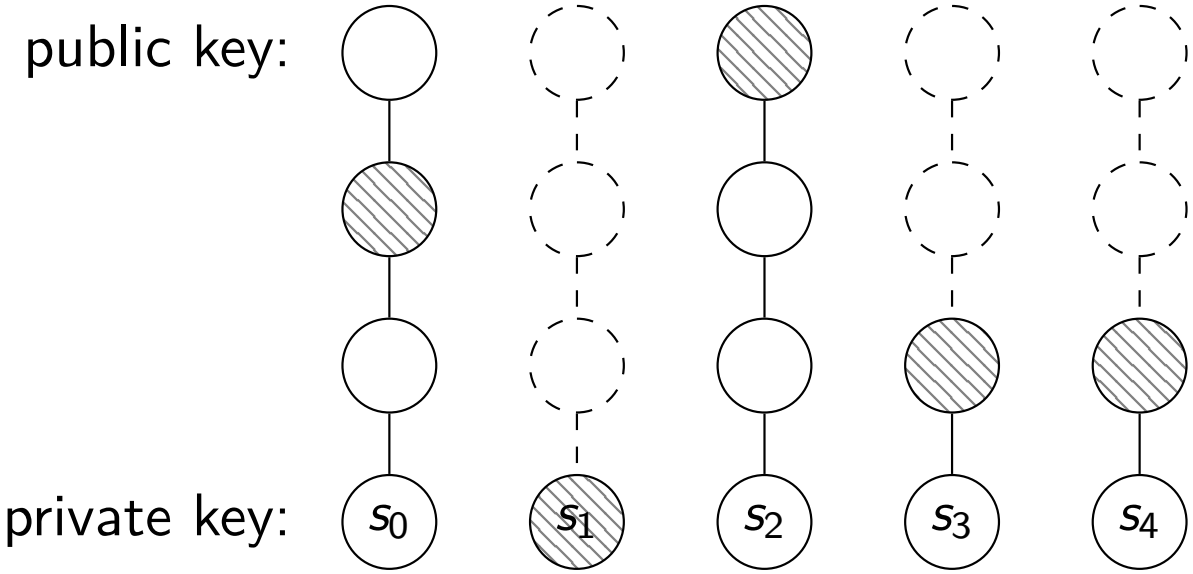
- Idea: sign groups of $\log(w)$ bits (let $w = 2^n$)
- Trade time for signature and key size
- Example: $w = 4$, let's sign `10 00 11 01 01`



Note: 'checksum chains' to prevent forgery omitted for simplicity

The Winternitz improvement

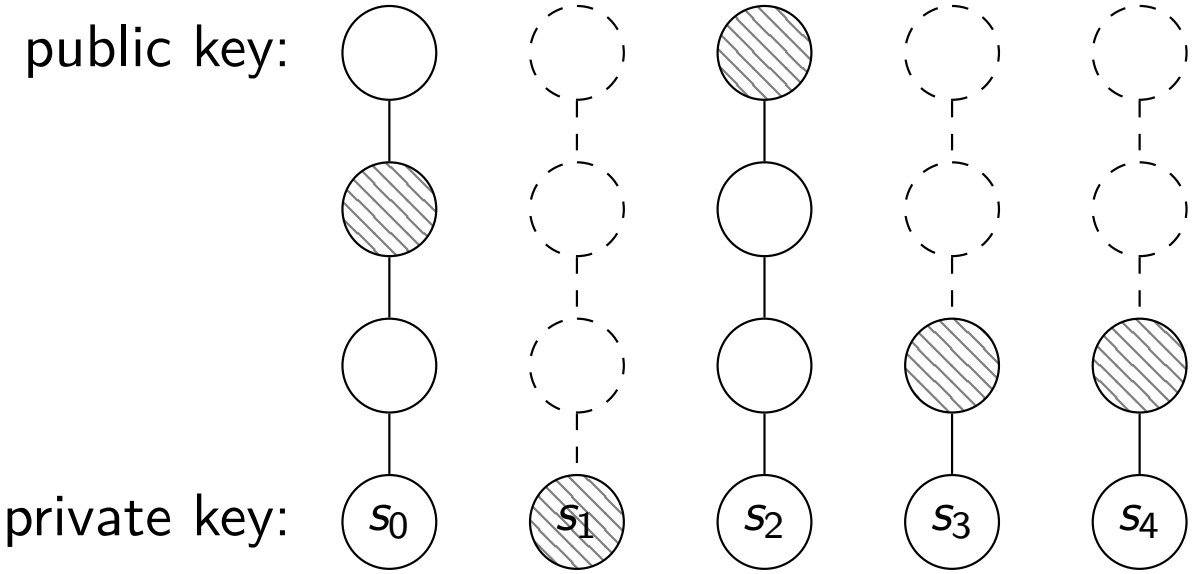
- Idea: sign groups of $\log(w)$ bits (let $w = 2^n$)
- Trade time for signature and key size
- Example: $w = 4$, let's sign `10 00 11 01 01`



Note: 'checksum chains' to prevent forgery omitted for simplicity

The Winternitz improvement

- Idea: sign groups of $\log(w)$ bits (let $w = 2^n$)
- Trade time for signature and key size
- Example: $w = 4$, let's sign `10 00 11 01 01`



- Can still only do this **once**!

Note: 'checksum chains' to prevent forgery omitted for simplicity

Merkle trees

- One public key, multiple signatures?
  - OTS, so multiple signatures $\rightarrow$ multiple private keys

Merkle trees

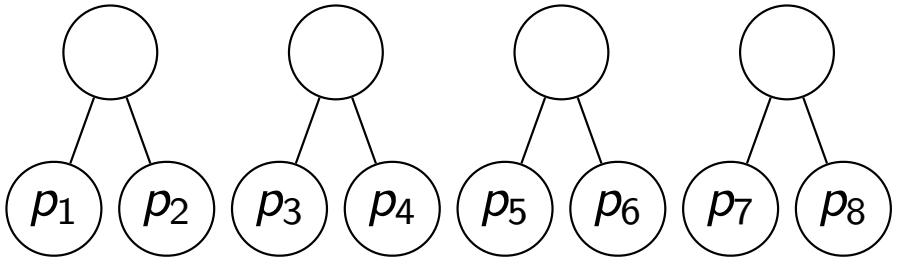
- One public key, multiple signatures?
  - OTS, so multiple signatures $\rightarrow$ multiple private keys
- Merkle: build 'authentication tree' on top

p_1 p_2 p_3 p_4 p_5 p_6 p_7 p_8

- Leaf p_i = OTS public key i

Merkle trees

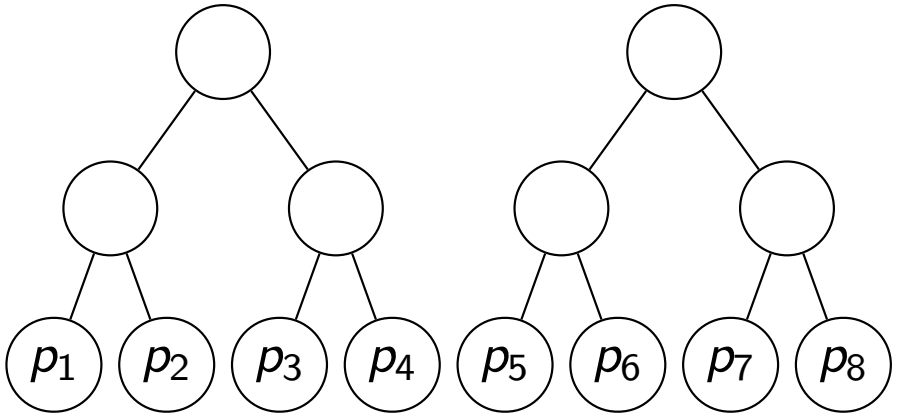
- One public key, multiple signatures?
  - OTS, so multiple signatures $\rightarrow$ multiple private keys
- Merkle: build 'authentication tree' on top



- Leaf p_i = OTS public key i
- Parent = h(LeftChild $\|$ RightChild)

Merkle trees

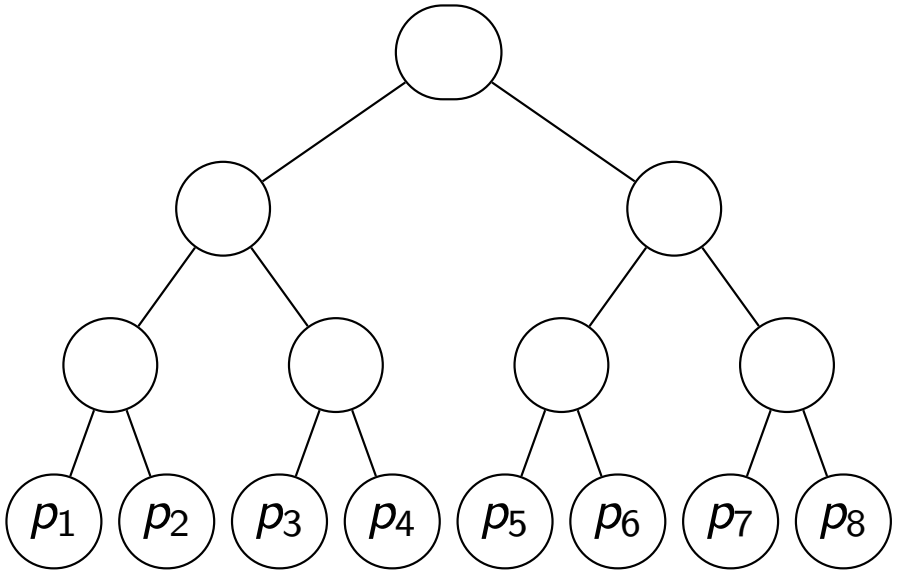
- One public key, multiple signatures?
  - OTS, so multiple signatures $\rightarrow$ multiple private keys
- Merkle: build 'authentication tree' on top



- Leaf p_i = OTS public key i
- Parent = h(LeftChild $\|$ RightChild)

Merkle trees

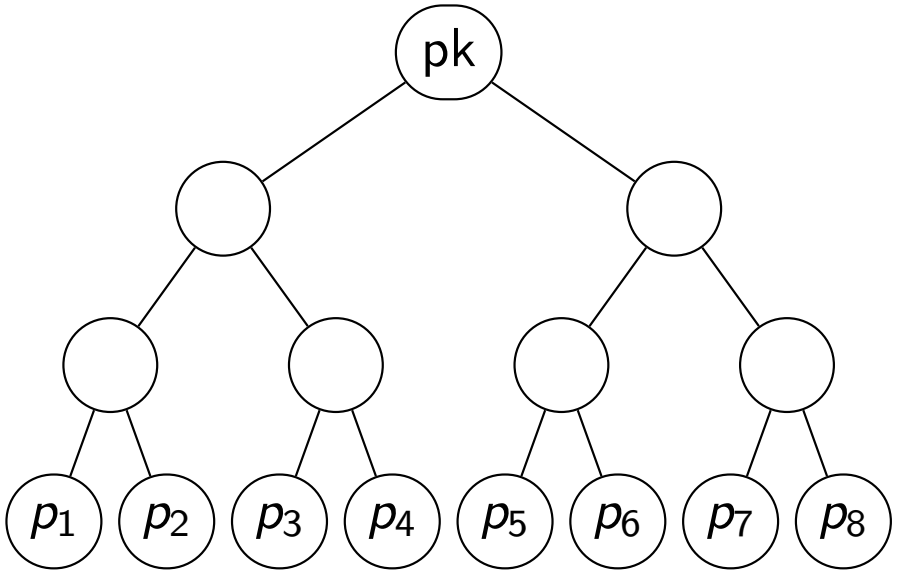
- One public key, multiple signatures?
  - OTS, so multiple signatures $\rightarrow$ multiple private keys
- Merkle: build 'authentication tree' on top



- Leaf p_i = OTS public key i
- Parent = h(LeftChild $\|$ RightChild)

Merkle trees

- One public key, multiple signatures?
  - OTS, so multiple signatures $\rightarrow$ multiple private keys
- Merkle: build 'authentication tree' on top



- Leaf p_i = OTS public key i
- Parent = h(LeftChild $\|$ RightChild)
- New public key: root node

Merkle trees

- Signature must now include:
  - OTS signature

Merkle trees

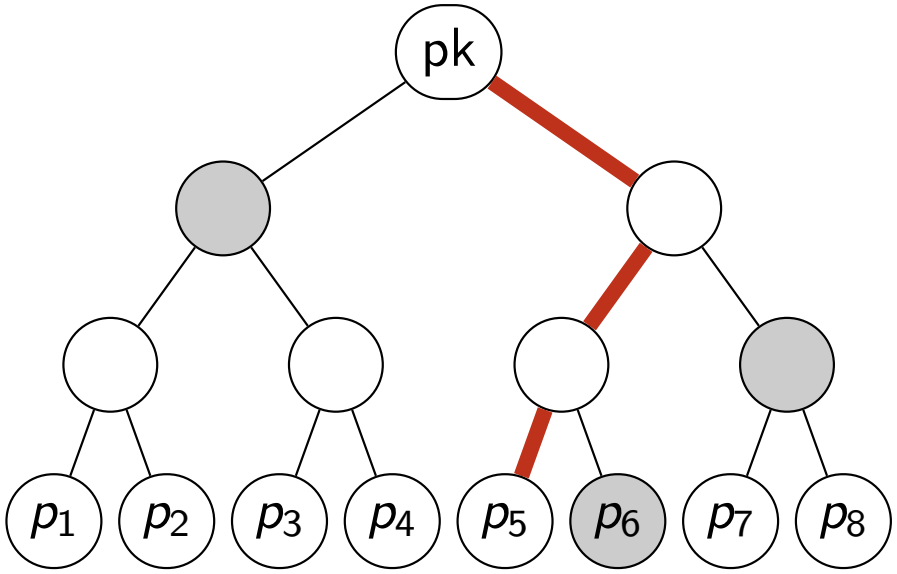
- Signature must now include:
  - OTS signature
  - OTS public key

Merkle trees

- Signature must now include:
  - OTS signature
  - OTS public key
  - Index in the Merkle tree, e.g. 5

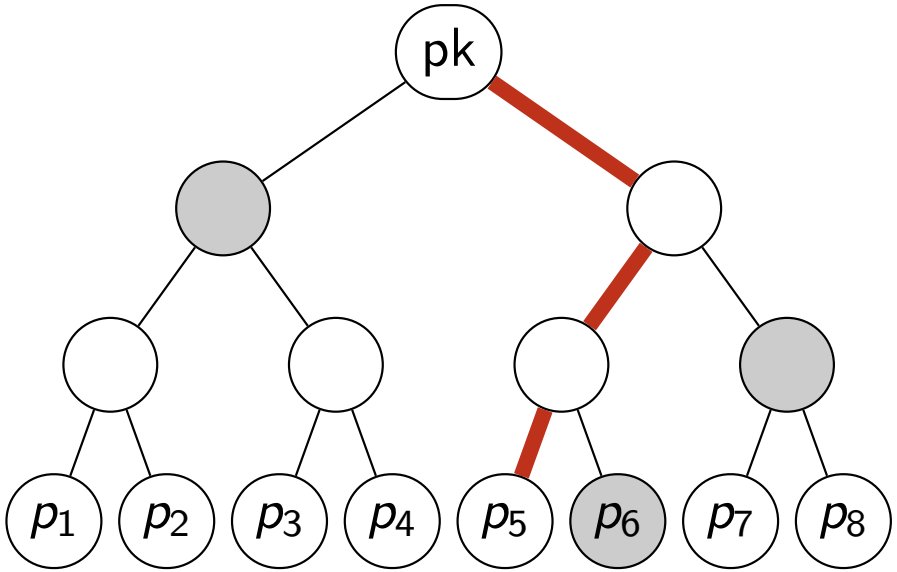
Merkle trees

- Signature must now include:
  - OTS signature
  - OTS public key
  - Index in the Merkle tree, e.g. 5
  - Nodes along the *authentication path*



Merkle trees

- Signature must now include:
  - OTS signature
  - OTS public key
  - Index in the Merkle tree, e.g. 5
  - Nodes along the *authentication path*



- Verification
  1. Implicitly verify OTS signature (reconstruct OTS public key)
  2. Reconstruct root node (using authentication path)

Analysis

- Say we do 2^{32} signatures ($\approx 4 \cdot 10^9$)

Analysis

- Say we do 2^{32} signatures ($\approx 4 \cdot 10^9$)
- Key generation is slow: compute tree of $2^{33} - 1$ nodes

Analysis

- Say we do 2^{32} signatures ($\approx 4 \cdot 10^9$)
- Key generation is slow: compute tree of $2^{33} - 1$ nodes
- Signing is fast

Analysis

- Say we do 2^{32} signatures ($\approx 4 \cdot 10^9$)
- Key generation is slow: compute tree of $2^{33} - 1$ nodes

- Signing is fast
  - OTS signature on (hash of) message
  - Small update to authentication path

Analysis

- Say we do 2^{32} signatures ($\approx 4 \cdot 10^9$)
- Key generation is slow: compute tree of $2^{33} - 1$ nodes

- Signing is fast
  - OTS signature on (hash of) message
  - Small update to authentication path
- Keys are small
  - Public key is one hash value

Analysis

- Say we do 2^{32} signatures ($\approx 4 \cdot 10^9$)
- Key generation is slow: compute tree of $2^{33} - 1$ nodes

- Signing is fast
  - OTS signature on (hash of) message
  - Small update to authentication path
- Keys are small
  - Public key is one hash value
  - Private key is billions of random values

Analysis

- Say we do 2^{32} signatures ($\approx 4 \cdot 10^9$)
- Key generation is slow: compute tree of $2^{33} - 1$ nodes

- Signing is fast
  - OTS signature on (hash of) message
  - Small update to authentication path
- Keys are small
  - Public key is one hash value
  - Private key is ~~billions of random values~~ a seed

Analysis

- Say we do 2^{32} signatures ($\approx 4 \cdot 10^9$)
- Key generation is slow: compute tree of $2^{33} - 1$ nodes

- Signing is fast
  - OTS signature on (hash of) message
  - Small update to authentication path
- Keys are small
  - Public key is one hash value
  - Private key is ~~billions of random values~~ a seed
- Signatures are small:
  - OTS signature (2 KiB) + authentication path (1 KiB)

Analysis

- Say we do 2^{32} signatures ($\approx 4 \cdot 10^9$)
- Key generation is slow: compute tree of $2^{33} - 1$ nodes

- Signing is fast
  - OTS signature on (hash of) message
  - Small update to authentication path
- Keys are small
  - Public key is one hash value
  - Private key is ~~billions of random values~~ a seed
- Signatures are small:
  - OTS signature (2 KiB) + authentication path (1 KiB)

- Can only use each leaf node **once**

Analysis

- Say we do 2^{32} signatures ($\approx 4 \cdot 10^9$)
- Key generation is slow: compute tree of $2^{33} - 1$ nodes

- Signing is fast
  - OTS signature on (hash of) message
  - Small update to authentication path
- Keys are small
  - Public key is one hash value
  - Private key is ~~billions of random values~~ a seed
- Signatures are small:
  - OTS signature (2 KiB) + authentication path (1 KiB)

- Can only use each leaf node **once**
  - We must **store and update** the index

Analysis

- Say we do 2^{32} signatures ($\approx 4 \cdot 10^9$)
- Key generation is slow: compute tree of $2^{33} - 1$ nodes

- Signing is fast
  - OTS signature on (hash of) message
  - Small update to authentication path
- Keys are small
  - Public key is one hash value
  - Private key is ~~billions of random values~~ a seed
- Signatures are small:
  - OTS signature (2 KiB) + authentication path (1 KiB)

- Can only use each leaf node **once**
  - We must **store and update** the index
  - **We must keep a state!**

ELIMINATE

THE STATE

ELIMINATE THE STATE

Special note to law-enforcement agents:
"The word 'state' is a technical term in cryptography. [..] We are not talking about eliminating other types of states. We love most states, especially yours! Also, 'hash' is another technical term and has nothing to do with cannabis."
— https://sphincs.cr.yp.to

SPHINCS

- *Seriously* big tree ($\approx 2^{64}$ leafs)
  $\Rightarrow$ Allows random leaf selection

Note: omitting bottom layer of 'Few-Time Signatures' for simplicity

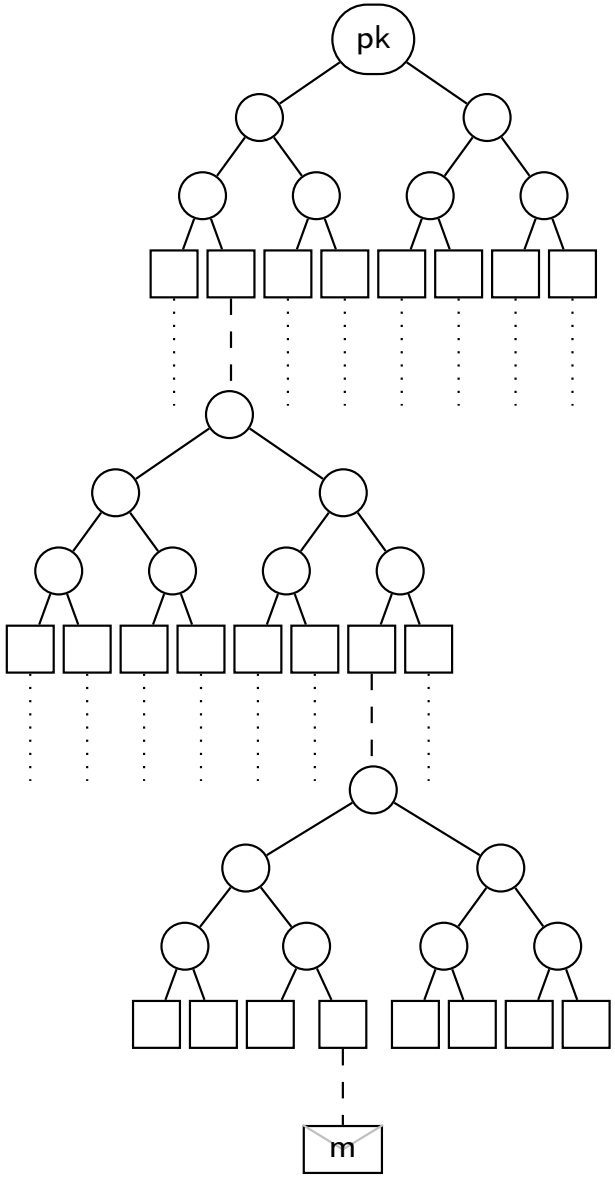
SPHINCS

- *Seriously* big tree ($\approx 2^{64}$ leafs)
  - $\Rightarrow$ Allows random leaf selection
  - $\Rightarrow$ Stateless!

Note: omitting bottom layer of 'Few-Time Signatures' for simplicity

SPHINCS

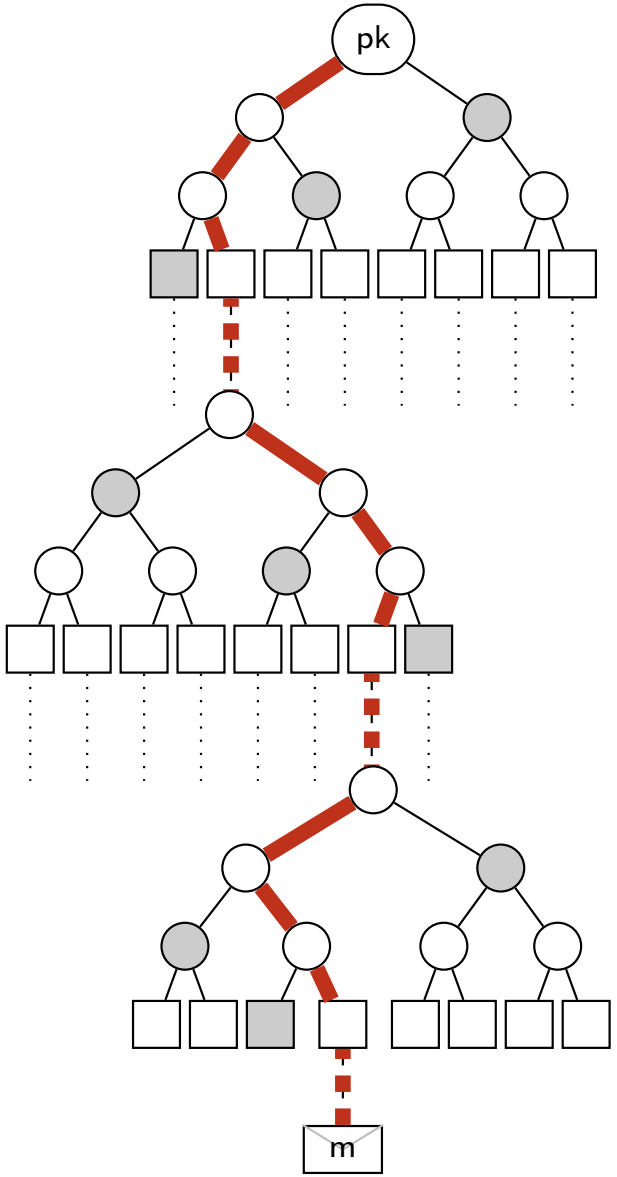
- *Seriously* big tree ($\approx 2^{64}$ leafs)
  - $\Rightarrow$ Allows random leaf selection
  - $\Rightarrow$ Stateless!
- Cannot generate entire tree!
  - 'Tree of trees'
  - Only generate needed subtrees
  - Link trees with OTS



Note: omitting bottom layer of 'Few-Time Signatures' for simplicity

SPHINCS

- *Seriously* big tree ($\approx 2^{64}$ leafs)
  - $\Rightarrow$ Allows random leaf selection
  - $\Rightarrow$ Stateless!
- Cannot generate entire tree!
  - 'Tree of trees'
  - Only generate needed subtrees
  - Link trees with OTS
- Signatures larger and slower
  - 8 KiB – 40 KiB, $\approx$ 100ms



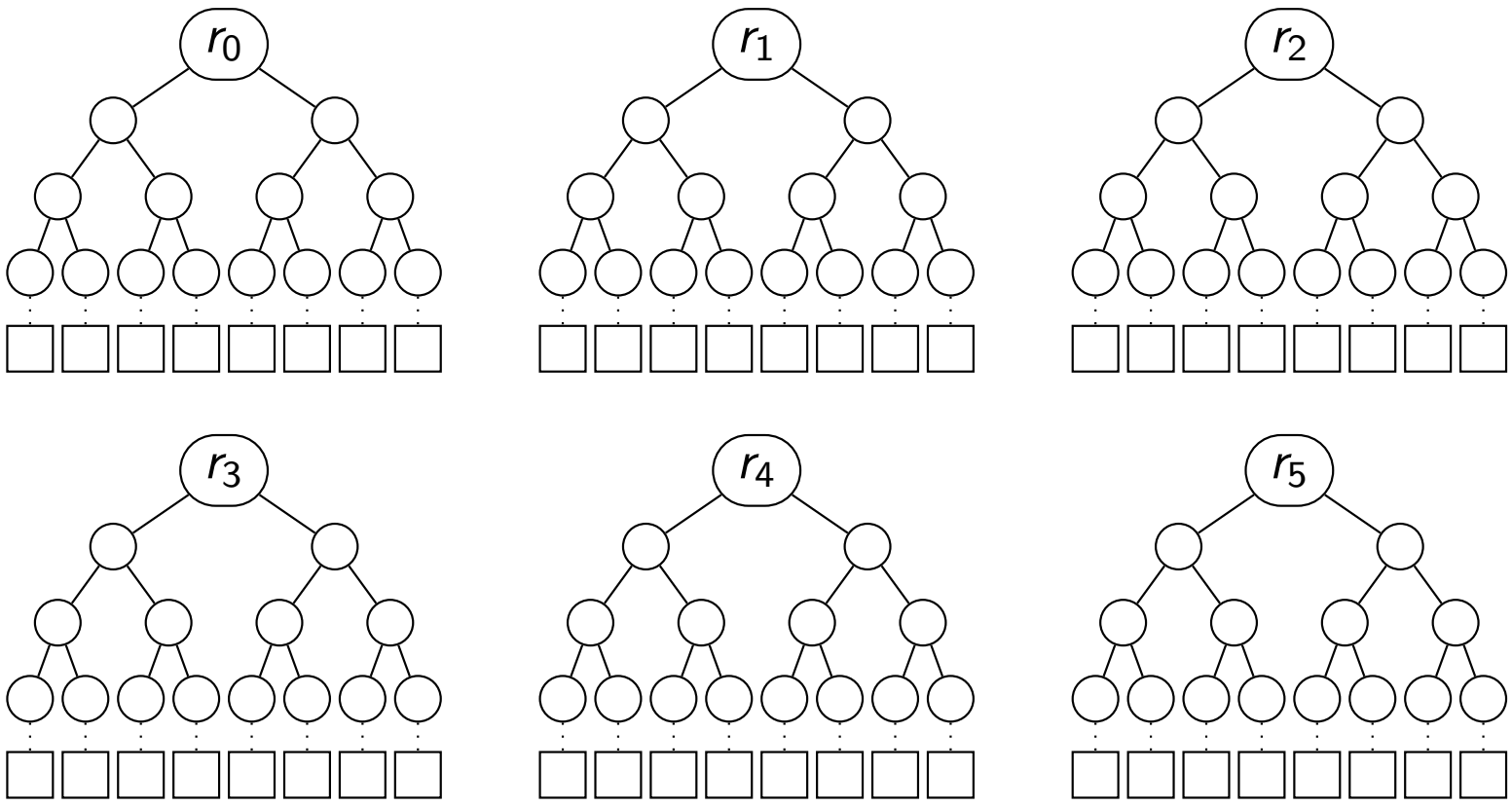
Note: omitting bottom layer of 'Few-Time Signatures' for simplicity

Forest of Random Subsets

- 'Few-time' signature scheme to sign $m \Rightarrow$ shorter hypertree

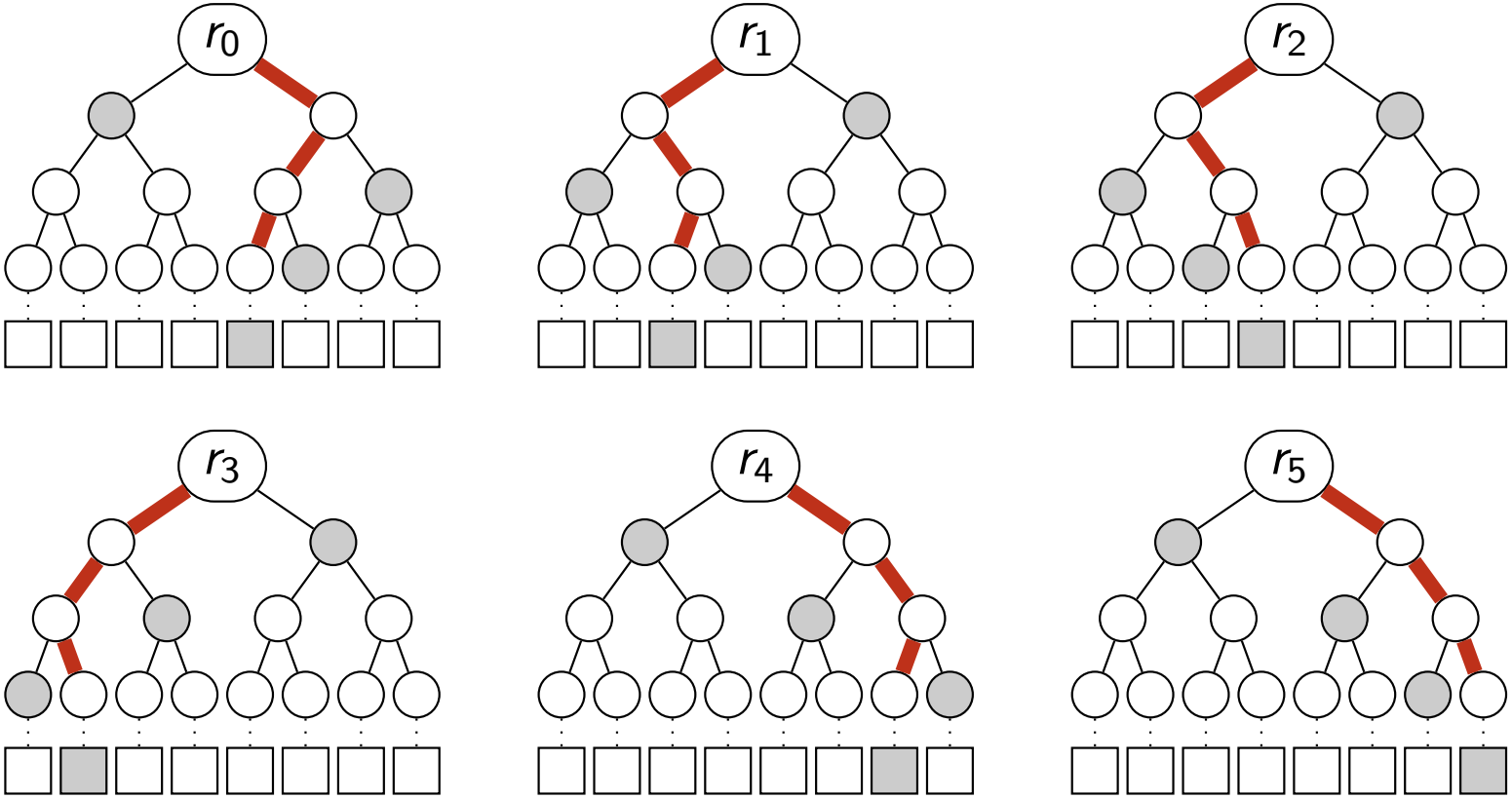
Forest of Random Subsets

- 'Few-time' signature scheme to sign $m \Rightarrow$ shorter hypertree
- Ex.: $d = 6$, $\log(t) = 3$, sign `100 010 011 001 110 111`



Forest of Random Subsets

- 'Few-time' signature scheme to sign $m \Rightarrow$ shorter hypertree
- Ex.: $d = 6$, $\log(t) = 3$, sign `100 010 011 001 110 111`



- Public key: $\mathrm{h}(r_0, r_1, \ldots, r_5)$
- Signature: 6x sk (■), 6x authentication path (●, ●, ●)

More of this?

- Year 1: Security ✓
- Year 2: Introduction to Cryptography (elective)
- TRU/e: Cryptology
- TRU/e: Cryptographic Engineering (elective)

- (Maths BSc: Rings & Fields)

More of this?

- Year 1: Security ✓
- Year 2: Introduction to Cryptography (elective)
- TRU/e: Cryptology
- TRU/e: Cryptographic Engineering (elective)

- (Maths BSc: Rings & Fields)

- Implement your own crypto!

More of this?

- Year 1: Security ✓
- Year 2: Introduction to Cryptography (elective)
- TRU/e: Cryptology
- TRU/e: Cryptographic Engineering (elective)

- (Maths BSc: Rings & Fields)

- Implement your own crypto!
  - ...but maaaybe don't use it in production

More of this?

- Year 1: Security ✓
- Year 2: Introduction to Cryptography (elective)
- TRU/e: Cryptology
- TRU/e: Cryptographic Engineering (elective)

- (Maths BSc: Rings & Fields)

- Implement your own crypto!
  - ...but maaaybe don't use it in production

- Ask questions!

References I

Daniel J. Bernstein, Diana Hopwood, Andreas Hülsing, Tanja Lange, Ruben Niederhagen, Louiza Papachristodoulou, Peter Schwabe and Zooko Wilcox O'Hearn.

SPHINCS: Stateless, practical, hash-based, incredibly nice cryptographic signatures.

In Marc Fischlin and Elisabeth Oswald, editors, *Advances in Cryptology – EUROCRYPT 2015*, volume 9056 of *LNCS*, pages 368-397. Springer, 2015.

https://eprint.iacr.org/2014/795.pdf

John Rompel.

One-way functions are necessary and sufficient for secure signatures.

In *Proceedings of the twenty-second annual ACM symposium on theory of computing*, pages 387–394. ACM, 1990.

https://www.cs.princeton.edu/courses/archive/spr08/cos598D/Rompel.pdf

References II

Ralph Merkle.

A certified digital signature.

In Gilles Brassard, editor, *Advances in Cryptology – Crypto '89*, volume 435 of *LNCS*, pages 218-238. Springer-Verlag, 1990.

http://www.merkle.com/papers/Certified1979.pdf

Oded Goldreich.

Two remarks concerning the Goldwasser-Micali-Rivest signature scheme.

In Andrew M. Odlyzko, editor, *Advances in Cryptology – Crypto '86*, volume 263 of *LNCS*, pages 104-110. Springer-Verlag, 1987.

http://www.wisdom.weizmann.ac.il/~oded/PSX/gmr.pdf

Andreas Hülsing.

W-OTS+ – shorter signatures for hash-based signature schemes.

In Amr Youssef, Abderrahmane Nitaj and Aboul-Ella Hassanien, editors, *Progress in Cryptology – AFRICACRYPT 2013*, volume 7918 of *LNCS*, pages 173-188. Springer, 2013.

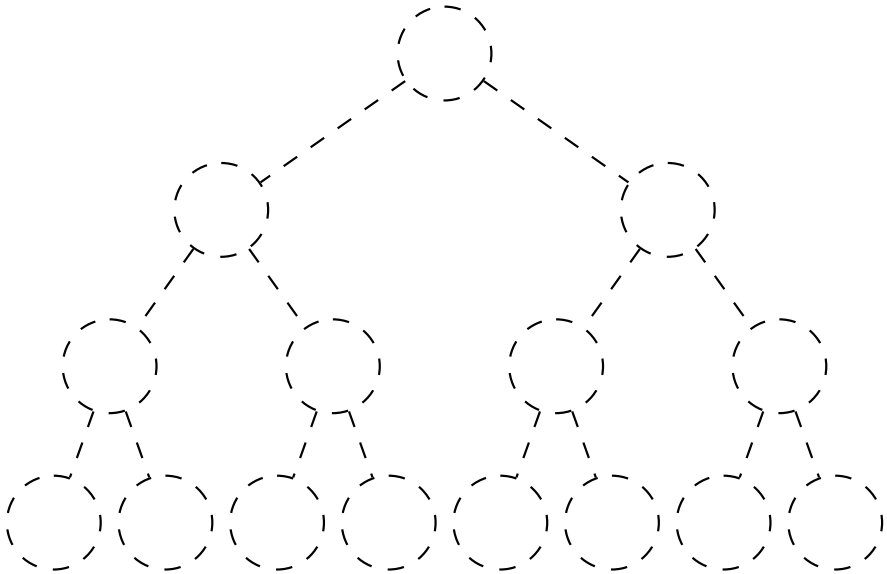
https://eprint.iacr.org/2017/965.pdf

Treehash

- Merkle trees are large!

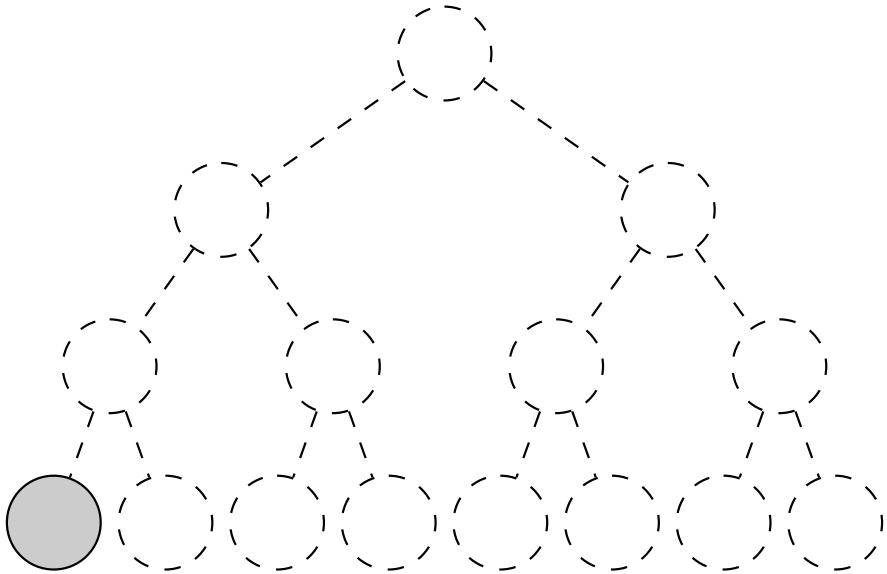
Treehash

- Merkle trees are large!
- Treehash: only remember relevant nodes
  - Maintain a stack: max. $\log(2^h)$ nodes



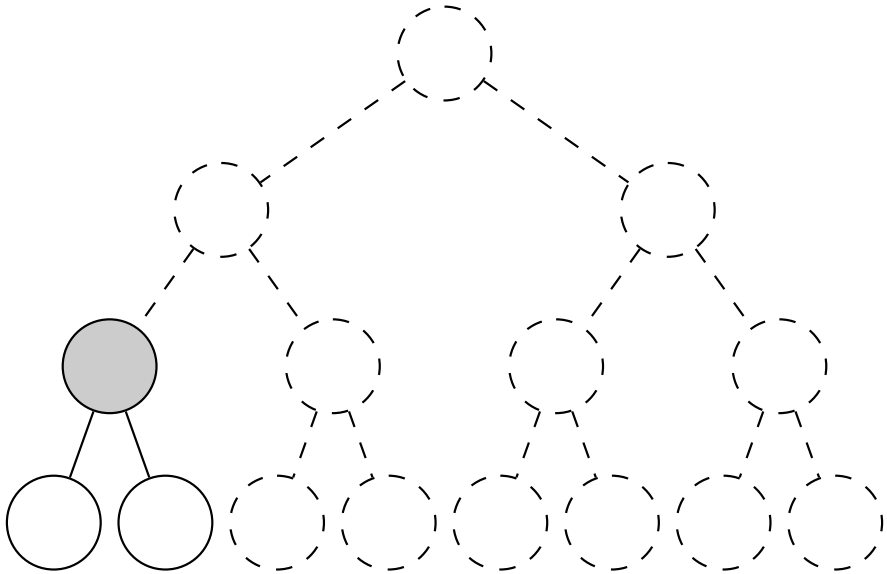
Treehash

- Merkle trees are large!
- Treehash: only remember relevant nodes
  - Maintain a stack: max. $\log(2^h)$ nodes



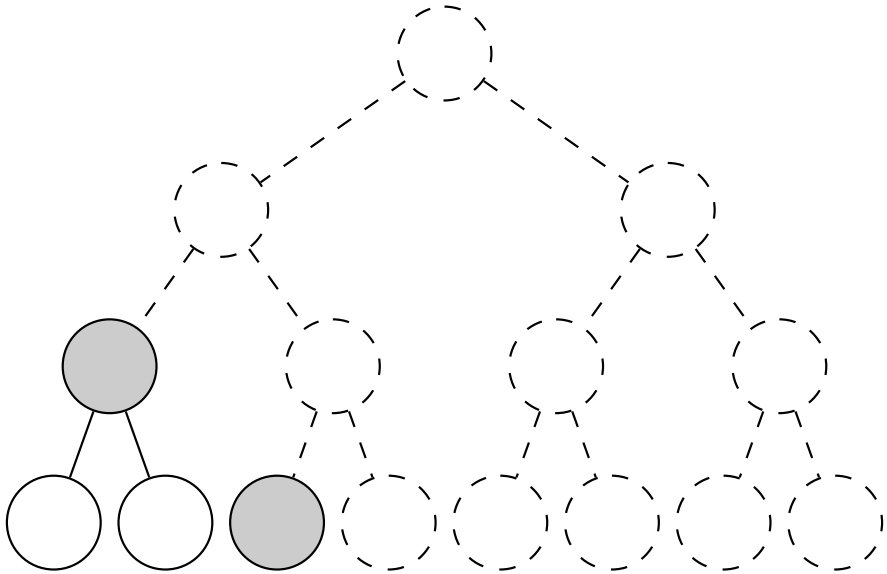
Treehash

- Merkle trees are large!
- Treehash: only remember relevant nodes
  - Maintain a stack: max. $\log(2^h)$ nodes



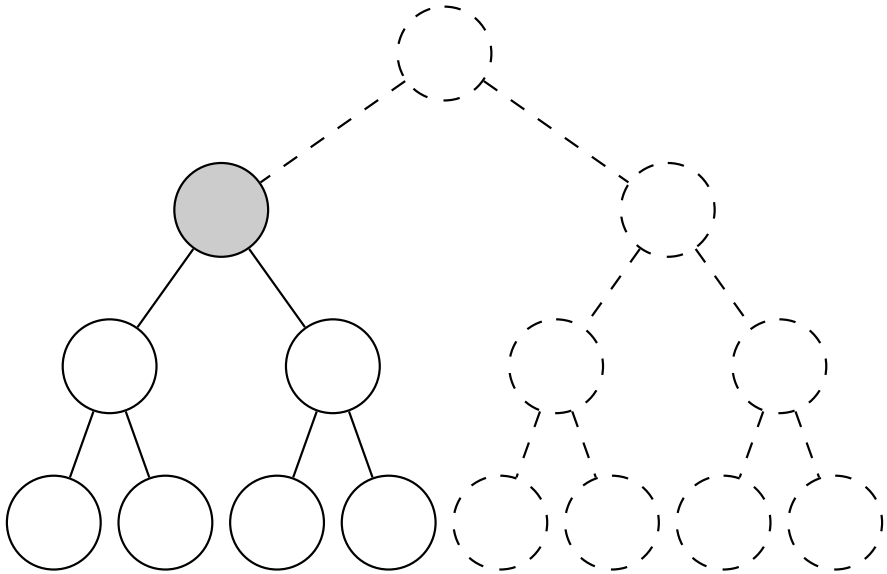
Treehash

- Merkle trees are large!
- Treehash: only remember relevant nodes
  - Maintain a stack: max. $\log(2^h)$ nodes



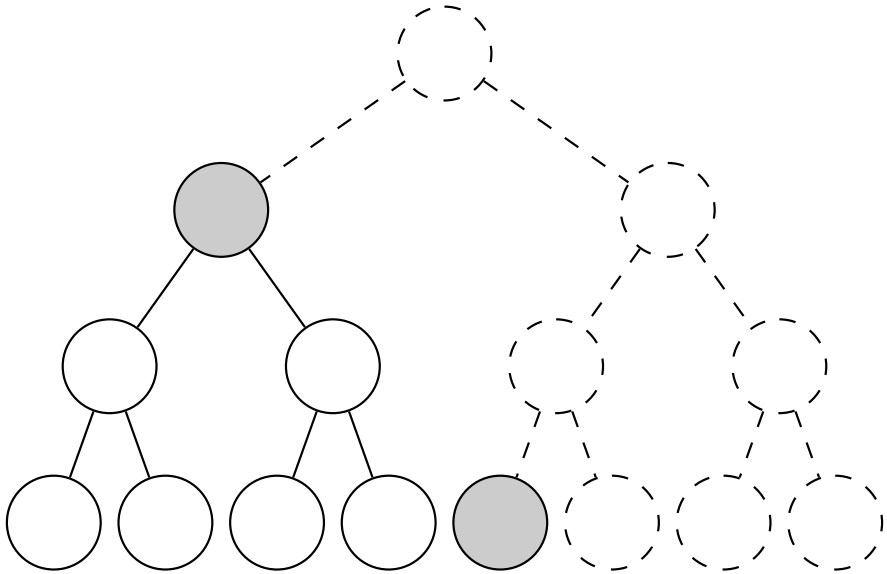
Treehash

- Merkle trees are large!
- Treehash: only remember relevant nodes
  - Maintain a stack: max. $\log(2^h)$ nodes



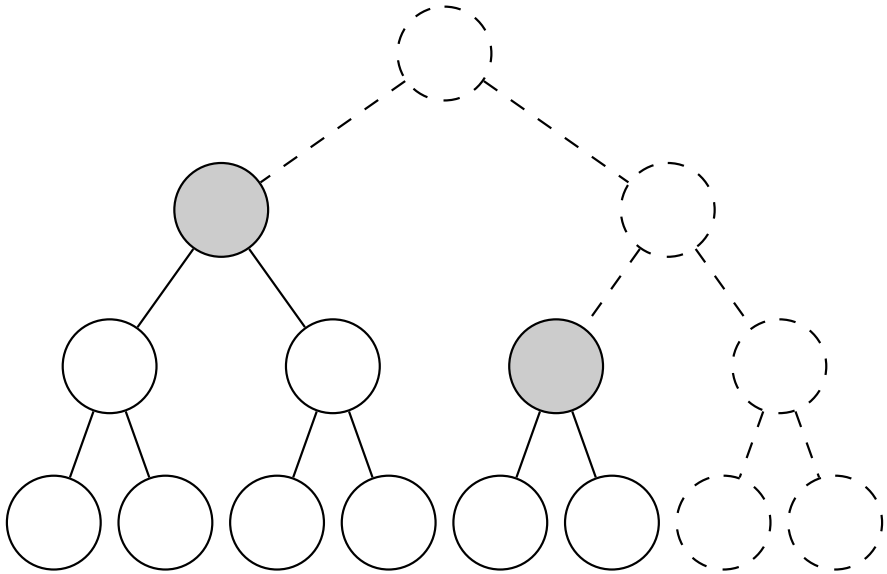
Treehash

- Merkle trees are large!
- Treehash: only remember relevant nodes
  - Maintain a stack: max. $\log(2^h)$ nodes



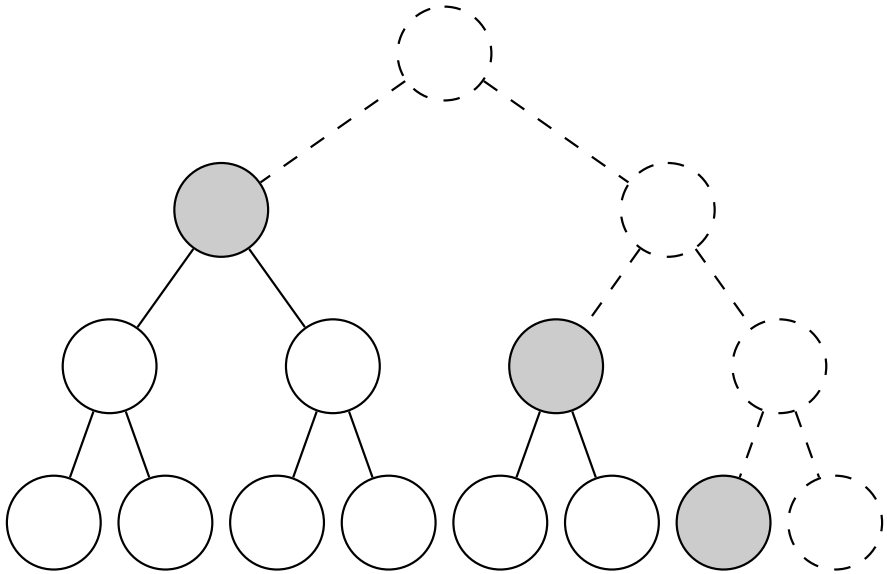
Treehash

- Merkle trees are large!
- Treehash: only remember relevant nodes
  - Maintain a stack: max. $\log(2^h)$ nodes



Treehash

- Merkle trees are large!
- Treehash: only remember relevant nodes
  - Maintain a stack: max. $\log(2^h)$ nodes



Treehash

- Merkle trees are large!
- Treehash: only remember relevant nodes
  - Maintain a stack: max. $\log(2^h)$ nodes