Post-quantum cryptography

Geovandro C. C. F. Pereira
Institute for Quantum Computing
University of Waterloo

Agenda

- Preliminaries and classical crypto issues
- Post-quantum requirements and candidates
- History on crypto standadization and current NIST standardization
- Hash-based cryptography

What is security after all?

Computer Security

Computer security is the **protection afforded** to an information system in order to **achieve** the applicable goals of **integrity/authenticity, availability**, and **confidentiality** of the resources (includes hardware, software, information/data, and telecommunications).

– adapted from NIST Computer Security Handbook

What is security after all?

Computer Security

Computer security is the **protection afforded** to an information system in order to **achieve** the applicable goals of **integrity/authenticity, availability**, and **confidentiality** of the resources (includes hardware, software, information/data, and telecommunications).
– adapted from NIST Computer Security Handbook

- Security requires more than confidentiality. Solutions are not supported by **information-theoretic security** and a tradeoff must be analized:

What is security after all?

Computer Security

Computer security is the **protection afforded** to an information system in order to **achieve** the applicable goals of **integrity/authenticity, availability**, and **confidentiality** of the resources (includes hardware, software, information/data, and telecommunications).
– adapted from NIST Computer Security Handbook

- Security requires more than confidentiality. Solutions are not supported by **information-theoretic security** and a tradeoff must be analized:
  - $\Uparrow$ security $\implies \Uparrow$ cost

What is security after all?

Computer Security

Computer security is the **protection afforded** to an information system in order to **achieve** the applicable goals of **integrity/authenticity, availability**, and **confidentiality** of the resources (includes hardware, software, information/data, and telecommunications).
– adapted from NIST Computer Security Handbook

- Security requires more than confidentiality. Solutions are not supported by **information-theoretic security** and a tradeoff must be analized:
  - $\Uparrow$ security $\Longrightarrow \Uparrow$ cost
  - Increase parameters: $\Uparrow$ bandwidth/energy consumption, $\Uparrow$ processing
  - Offer rewards for bug hunters

Security Goals (CIA, not the agency)

- **C**onfidentiality (symmetric): prevent eavesdropping

Security Goals (CIA, not the agency)

- **C**onfidentiality (symmetric): prevent eavesdropping
  - instant messaging, local storage

Security Goals (CIA, not the agency)

- **C**onfidentiality (symmetric): prevent eavesdropping
  - instant messaging, local storage
  - symmetric ciphers: AES (block cipher), ChaCha20 (stream cipher) ...

Security Goals (CIA, not the agency)

- **I**ntegrity (symmetric): prevent data tampering

Security Goals (CIA, not the agency)

- **I**ntegrity (symmetric): prevent data tampering
  - control messages: pacemakers, drones

Security Goals (CIA, not the agency)

- **I**ntegrity (symmetric): prevent data tampering
  - control messages: pacemakers, drones

  - Hash functions, MACs (tag): SHA-2, HMAC (keyed hash), Poly1305.

Security Goals (CIA, not the agency)

- **A**uthenticity (asymmetric): check the origin

Security Goals (CIA, not the agency)

- **A**uthenticity (asymmetric): check the origin
  - web browsing, e-banking, sw updates

Security Goals (CIA, not the agency)

- **A**uthenticity (asymmetric): check the origin
  - web browsing, e-banking, sw updates
  - Public key crypto: RSA, (EC)DSA, ...

Quantum cryptanalysis effects

Impact on public-key (asymmetric) cryptography

- Shor 1994: *algorithms for quantum computation: discrete logarithms and factoring*
  - ✗ No more RSA and DSA (discrete log-based)
- ✗ 2003, extension to ECDSA Proos and Zalka 2003 UNIVERSITY OF WATERLOO

Quantum cryptanalysis effects

Impact on public-key (asymmetric) cryptography

- Shor 1994: *algorithms for quantum computation: discrete logarithms and factoring*
  - ✗ No more RSA and DSA (discrete log-based)
- ✗ 2003, extension to ECDSA Proos and Zalka 2003

of qubits required

- **Factoring**: $2n + 2$ logical qubits[a]
  - RSA-3072: 6146 qubits

[a] Häner, Roetteler, and Svore 2016.
[b] Roetteler et al. 2017.

Quantum cryptanalysis effects

Impact on public-key (asymmetric) cryptography

- Shor 1994: *algorithms for quantum computation: discrete logarithms and factoring*
  - ✗ No more RSA and DSA (discrete log-based)
- ✗ 2003, extension to ECDSA Proos and Zalka 2003

of qubits required

- **Factoring**: $2n + 2$ logical qubits[a]
  - RSA-3072: 6146 qubits
- **ECDLP**: $9n + 2\lceil \log n \rceil + 10$ logical qubits[b]
  - ECC-256: 2330 qubits

[a] Häner, Roetteler, and Svore 2016.
[b] Roetteler et al. 2017.

Quantum cryptanalysis effects

Impact on public-key (asymmetric) cryptography

- Shor 1994: *algorithms for quantum computation: discrete logarithms and factoring*
  - ✗ No more RSA and DSA (discrete log-based)
- ✗ 2003, extension to ECDSA Proos and Zalka 2003

of qubits required

- **Factoring**: $2n + 2$ logical qubits[a]
  - RSA-3072: 6146 qubits
- **ECDLP**: $9n + 2\lceil \log n \rceil + 10$ logical qubits[b]
  - ECC-256: 2330 qubits
- For 128-bit security "ECC easier target than RSA"[b].

[a] Häner, Roetteler, and Svore 2016.

[b] Roetteler et al. 2017.

Quantum cryptanalysis effects

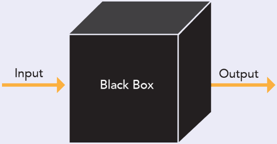
Impact on symmetric crypto

- Grover 1996: *a fast quantum mechanical algorithm for database search*

Quantum cryptanalysis effects

Impact on symmetric crypto

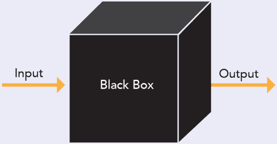
- Grover 1996: *a fast quantum mechanical algorithm for database search*
- **unstructured search**: find the unique input to black box function for a given output



Quantum cryptanalysis effects

Impact on symmetric crypto

- Grover 1996: *a fast quantum mechanical algorithm for database search*
- **unstructured search**: find the unique input to black box function for a given output

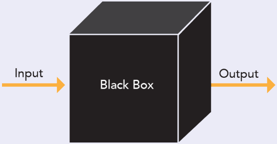


- Complexity is **optimal** Bennett et al. 1997:
  No hope of solving NP-complete problems: $o(\sqrt{N})$ lower bound.

Quantum cryptanalysis effects

Impact on symmetric crypto

- Grover 1996: *a fast quantum mechanical algorithm for database search*
- **unstructured search**: find the unique input to black box function for a given output

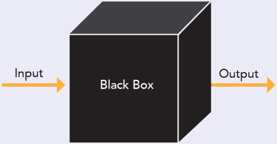


- Complexity is **optimal** Bennett et al. 1997:
  No hope of solving NP-complete problems: $o(\sqrt{N})$ lower bound.
- Partially affects block-ciphers.

Quantum cryptanalysis effects

Impact on symmetric crypto

- Grover 1996: *a fast quantum mechanical algorithm for database search*
- **unstructured search**: find the unique input to black box function for a given output



- Complexity is **optimal** Bennett et al. 1997:
  No hope of solving NP-complete problems: $o(\sqrt{N})$ lower bound.
- Partially affects block-ciphers.
  - Easy countermeasure: double up key sizes.

What are the affected families of cryptosystems?

- **C**onfidentiality, **I**ntegrity (symmetric) ✓

Model of block ciphers

If the encryption key is chosen at random, then an attacker who does not know the key cannot distinguish between the block cipher and a truly random permutation.

$$f(X) = E_X(I)$$

What are the affected families of cryptosystems?

- **C**onfidentiality, **I**ntegrity (symmetric) ✓

Model of block ciphers

If the encryption key is chosen at random, then an attacker who does not know the key cannot distinguish between the block cipher and a truly random permutation.

$$f(X) = E_X(I)$$

- Moreover, hash functions are modeled as one-way functions.

What are the affected families of cryptosystems?

- **C**onfidentiality, **I**ntegrity (symmetric) ✓

Model of block ciphers

If the encryption key is chosen at random, then an attacker who does not know the key cannot distinguish between the block cipher and a truly random permutation.

$$f(X) = E_X(I)$$

- Moreover, hash functions are modeled as one-way functions.
- **A**uthentication (asymmetric) ✗

Assumptions do not hold in a quantum setting

Mainly based on the hardness of integer factoring or computing discrete logarithm.

But there is a hope

- Fortunately, some old not very interesting algorithms surprisingly resisted to Shor's new attacks.

But there is a hope

- Fortunately, some old not very interesting algorithms surprisingly resisted to Shor's new attacks.

Definition

Post-quantum cryptography consists of classical cryptographic algorithms whose security assumption does not suffer an exponential speedup by quantum attacks are automatically dubbed post-quantum algorithms.

But there is a hope

- Fortunately, some old not very interesting algorithms surprisingly resisted to Shor's new attacks.

Definition

Post-quantum cryptography consists of classical cryptographic algorithms whose security assumption does not suffer an exponential speedup by quantum attacks are automatically dubbed post-quantum algorithms.

Definition

Quantum-safe cryptography includes a broader set of cryprographic algorithms including non-classical assumptions such as laws of quantum physics, e.g. Quantum Key Distribution.

Post-quantum security

Essential security requirements for PQ-security:

1. The underlying computational problem must be hard on a quantum computer

Post-quantum security

Essential security requirements for PQ-security:

1. The underlying computational problem must be hard on a quantum computer
2. The security reduction from the cryptosystem to the hard problem must hold against quantum adversaries.

- Security proofs for classical Full-domain hash RSA or the Fiat-Shamir do not hold for post-quantum security.

Post-quantum security

Essential security requirements for PQ-security:

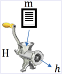
1. The underlying computational problem must be hard on a quantum computer
2. The security reduction from the cryptosystem to the hard problem must hold against quantum adversaries.

- Security proofs for classical Full-domain hash RSA or the Fiat-Shamir do not hold for post-quantum security.
  - Do not assume adversary can query an oracle $H(m)$ in superposition.

Post-quantum algorithms and security assumptions (beliefs)

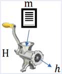
Hash-based crypto:

One-way functions exist, Lamport-Diffie: 40 years



Hash-based crypto:

One-way functions exist, Lamport-Diffie: 40 years



Multivariate-quadratic crypto:

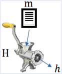
Random-looking multivariate system of non-linear equations is hard, UOV: 18 years

$$y_1 = x_1^2 + x_1x_2 + x_1x_4 + x_3$$
$$y_2 = x_3^2 + x_2x_3 + x_2x_4 + x_1 + 1$$
$$y_3 = \ldots$$

Post-quantum algorithms and security assumptions (beliefs)

Hash-based crypto:

One-way functions exist, Lamport-Diffie: 40 years



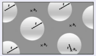
Multivariate-quadratic crypto:

Random-looking multivariate system of non-linear equations is hard, UOV: 18 years

$$y_1 = x_1^2 + x_1x_2 + x_1x_4 + x_3$$
$$y_2 = x_3^2 + x_2x_3 + x_2x_4 + x_1 + 1$$
$$y_3 = \ldots$$

Code-based crypto:

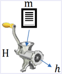
Syndrome decoding for error-correcting codes is hard, Goppa: 40 years



Post-quantum algorithms and security assumptions (beliefs)

Hash-based crypto:

One-way functions exist, Lamport-Diffie: 40 years



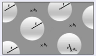
Multivariate-quadratic crypto:

Random-looking multivariate system of non-linear equations is hard, UOV: 18 years

$$y_1 = x_1^2 + x_1x_2 + x_1x_4 + x_3$$
$$y_2 = x_3^2 + x_2x_3 + x_2x_4 + x_1 + 1$$
$$y_3 = \ldots$$

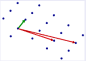
Code-based crypto:

Syndrome decoding for error-correcting codes is hard, Goppa: 40 years



Lattice-based crypto:

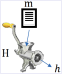
Finding short/close vectors on a lattice is hard, NTRU: 21 years, LWE: 12 years



Post-quantum algorithms and security assumptions (beliefs)

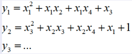
Hash-based crypto:

One-way functions exist, Lamport-Diffie: 40 years



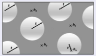
Multivariate-quadratic crypto:

Random-looking multivariate system of non-linear equations is hard, UOV: 18 years



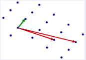
Code-based crypto:

Syndrome decoding for error-correcting codes is hard, Goppa: 40 years



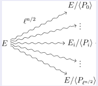
Lattice-based crypto:

Finding short/close vectors on a lattice is hard, NTRU: 21 years, LWE: 12 years



Supersingular Isogeny-based crypto:

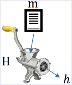
Computing isogenies between supersingular elliptic curves is hard, CGL hash: 12 years, SIDH kex: 6 years



Post-quantum algorithms and efficiency

Hash-based crypto:

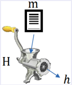
- Pros: simplicity (hash), efficient, high confidence
- Cons: large signatures (1–40 KiB), only signatures



Post-quantum algorithms and efficiency

Hash-based crypto:

- Pros: simplicity (hash), efficient, high confidence
- Cons: large signatures (1–40 KiB), only signatures



Multivariate-quadratic crypto:

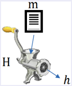
- Pros: small finite fields, matrix-vector ops, fast
- Cons: large keys (>10 KiB), **only signatures** safe

$$y_1 = x_1^2 + x_1x_2 + x_1x_4 + x_3$$
$$y_2 = x_3^2 + x_2x_3 + x_2x_4 + x_1 + 1$$
$$y_3 = \ldots$$

Post-quantum algorithms and efficiency

Hash-based crypto:

- Pros: simplicity (hash), efficient, high confidence
- Cons: large signatures (1–40 KiB), only signatures



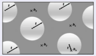
Multivariate-quadratic crypto:

- Pros: small finite fields, matrix-vector ops, fast
- Cons: large keys (>10 KiB), **only signatures** safe

$$y_1 = x_1^2 + x_1x_2 + x_1x_4 + x_3$$
$$y_2 = x_3^2 + x_2x_3 + x_2x_4 + x_1 + 1$$
$$y_3 = \dots$$

Code-based crypto:

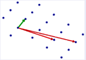
- Pros: small finite fields, matrix-vector ops, fast
- Cons: large keys (MiB), **only encryption** shown safe



Post-quantum algorithms and efficiency

Lattice-based crypto:

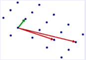
- Pros: versatile, moderate finite fields sizes, matrix/vector operations, very fast
- Cons: moderate key sizes (few KiB), practical parameters issues



Post-quantum algorithms and efficiency

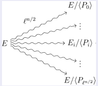
Lattice-based crypto:

- Pros: versatile, moderate finite fields sizes, matrix/vector operations, very fast
- Cons: moderate key sizes (few KiB), practical parameters issues



Supersingular Isogeny-based crypto:

- Pros: Small key sizes (few hundred bytes), many primitives
- Cons: relatively slower



The risk of sticking to pre-quantum crypto

- In August 2015, NSA announces that it plans to replace Suite B with a new cipher suite due to concerns about quantum computing attacks on ECC (even 256-bit curves).

The risk of sticking to pre-quantum crypto

- In August 2015, NSA announces that it plans to replace Suite B with a new cipher suite due to concerns about quantum computing attacks on ECC (even 256-bit curves).
- 2015, Koblitz N., Menezes A. *A riddle wrapped in an enigma*

The risk of sticking to pre-quantum crypto

- In August 2015, NSA announces that it plans to replace Suite B with a new cipher suite due to concerns about quantum computing attacks on ECC (even 256-bit curves).
- 2015, Koblitz N., Menezes A. *A riddle wrapped in an enigma*
- 2016, NIST announces post-quantum standardization
  - Deadline for submission Nov 30th, 2017
  - Focus on digital sigs, encryption and key establishment.

The risk of sticking to pre-quantum crypto

- In August 2015, NSA announces that it plans to replace Suite B with a new cipher suite due to concerns about quantum computing attacks on ECC (even 256-bit curves).
- 2015, Koblitz N., Menezes A. *A riddle wrapped in an enigma*
- 2016, NIST announces post-quantum standardization
  - Deadline for submission Nov 30th, 2017
  - Focus on digital sigs, encryption and key establishment.
- 2018, NIST PQC Standardization Conference (submitter's talks)

The risk of sticking to pre-quantum crypto

Mosca's risk analysis formula

Let X be the time to have certain information protected.
Let Y be the time to deploy post-quantum.
Let Z be the Y2Q (countdown of years to quantum).
If $Z < X + Y$: trouble!

What can one expect from Z?

- Huge investments in Quantum Computing research

NATURE | NEWS

Europe plans giant billion-euro quantum technologies project

Third European Union flagship will be similar in size and ambition to graphene and human brain initiatives.

Elizabeth Gibney

21 April 2016 | Updated: 26 April 2016

What can one expect from Z?

- Huge investments in Quantum Computing research

NATURE | NEWS

Europe plans giant billion-euro quantum technologies project

Third European Union flagship will be similar in size and ambition to graphene and human brain initiatives.

Elizabeth Gibney

21 April 2016 | Updated: 26 April 2016

- Industry competing for quantum supremacy (no crypto purpose).
  - 2017, Google and IBM building general-purpose small prototypes of QCs. Google has no fault-tolerance design plans.

What can one expect from Z?

- Huge investments in Quantum Computing research

NATURE | NEWS

Europe plans giant billion-euro quantum technologies project

Third European Union flagship will be similar in size and ambition to graphene and human brain initiatives.

Elizabeth Gibney

21 April 2016 | Updated: 26 April 2016

- Industry competing for quantum supremacy (no crypto purpose).
  - 2017, Google and IBM building general-purpose small prototypes of QCs. Google has no fault-tolerance design plans.
- More importantly, there is a steady progress in qubit fidelities. Experts estimate that large QCs (1k's of qubits) will be around by 2031 with 50% chance.

What can one expect from Y?

Y: Time to deploy new cryptography with wide interoperation

What can one expect from Y?

Y: Time to deploy new cryptography with wide interoperation

- Let's look at the history of ECC:

What can one expect from Y?

Y: Time to deploy new cryptography with wide interoperation

- Let's look at the history of ECC:
  - 1985: Elliptic curves in cryptography by Koblitz and Miller

What can one expect from Y?

Y: Time to deploy new cryptography with wide interoperation

- Let's look at the history of ECC:
  - 1985: Elliptic curves in cryptography by Koblitz and Miller
  - **1992**: ECDSA signature suggested by Scott Vanstone to DSS

What can one expect from Y?

Y: Time to deploy new cryptography with wide interoperation

- Let's look at the history of ECC:
  - 1985: Elliptic curves in cryptography by Koblitz and Miller
  - **1992**: ECDSA signature suggested by Scott Vanstone to DSS
  - 1996: Paul Kotcher introduces side-channel attacks.

What can one expect from Y?

Y: Time to deploy new cryptography with wide interoperation

- Let's look at the history of ECC:
  - 1985: Elliptic curves in cryptography by Koblitz and Miller
  - **1992**: ECDSA signature suggested by Scott Vanstone to DSS
  - 1996: Paul Kotcher introduces side-channel attacks.
    - 1997: Joye-Quisquater introduce fault attacks against ECC

What can one expect from Y?

Y: Time to deploy new cryptography with wide interoperation

- Let's look at the history of ECC:
  - 1985: Elliptic curves in cryptography by Koblitz and Miller
  - **1992**: ECDSA signature suggested by Scott Vanstone to DSS
  - 1996: Paul Kotcher introduces side-channel attacks.
    - 1997: Joye-Quisquater introduce fault attacks against ECC
    - 1998: The ECDLP is actually easy in some (anomalous) curves

What can one expect from Y?

Y: Time to deploy new cryptography with wide interoperation

- Let's look at the history of ECC:
  - 1985: Elliptic curves in cryptography by Koblitz and Miller
  - **1992**: ECDSA signature suggested by Scott Vanstone to DSS
  - 1996: Paul Kotcher introduces side-channel attacks.
    - 1997: Joye-Quisquater introduce fault attacks against ECC
    - 1998: The ECDLP is actually easy in some (anomalous) curves
    - 2000: Biehl-Meyer-Muller devise invalid curve attacks

What can one expect from Y?

Y: Time to deploy new cryptography with wide interoperation

- Let's look at the history of ECC:
  - 1985: Elliptic curves in cryptography by Koblitz and Miller
  - **1992**: ECDSA signature suggested by Scott Vanstone to DSS
  - 1996: Paul Kotcher introduces side-channel attacks.
    - 1997: Joye-Quisquater introduce fault attacks against ECC
    - 1998: The ECDLP is actually easy in some (anomalous) curves
    - 2000: Biehl-Meyer-Muller devise invalid curve attacks
  - **2000**: NIST FIPS 186-2 includes ECDSA and the 15 NIST curves

What can one expect for Y?

- Let's look at the history of ECC (cont.):

2014 Cloudfare's post dedicated to Scott Vanstone

W.r.t. **https** certificates, despite ECDSA being much faster than RSA for TLS handshake/signing, $>$ **90% of the certificates used on the web in 2014 were RSA-based**.
Even as late as 2012, out of 13 million TLS certificates found in a scan of the internet, fewer than 50 use an ECDSA key pair.

What can one expect for Y?

- Let's look at the history of ECC (cont.):

2014 Cloudfare's post dedicated to Scott Vanstone

W.r.t. **https** certificates, despite ECDSA being much faster than RSA for TLS handshake/signing, $>$ **90% of the certificates used on the web in 2014 were RSA-based**.
Even as late as 2012, out of 13 million TLS certificates found in a scan of the internet, fewer than 50 use an ECDSA key pair.

Main reason

Web sites owners slow to adopt new certificates due to maintainance of **compatibility with legacy browsers** that do not support the new algorithms. – Sullivan, N. 2014

What can one expect for Y?

The good news to ECC

In **Apr'17**, ECDSA finally **surpassed RSA** with 60% of all TLS connections. Recall that ECDSA was proposed in 1992 for DSS. It took 25 years for ECDSA to become widely deployed. – Cloudfare report.

What can one expect for Y?

The good news to ECC

In **Apr'17**, ECDSA finally **surpassed RSA** with 60% of all TLS connections. Recall that ECDSA was proposed in 1992 for DSS. It took 25 years for ECDSA to become widely deployed. – Cloudfare report.

The bad news to ECC

"For those partners and vendors that have not yet made the transition to Suite B algorithms, we recommend **not** making a significant expenditure to do so at this point but instead to prepare for the upcoming quantum resistant algorithm transition." – NSA 2015 announcement

What can one expect for Y?

Jul'17, Pereira's suggested bound:

Conjectured bound $Y \geq 25$

If $Z < X + 25 \Rightarrow$ trouble!

What can one expect for Y?

Jul'17, Pereira's suggested bound:

$$\text{Conjectured bound } Y \geq 25$$
$$\text{If } Z < X + 25 \Rightarrow \text{trouble!}$$

- It is true that the more robust TLS infrastructure and experienced commmunity will be faster at deploying implementations.

What can one expect for Y?

Jul'17, Pereira's suggested bound:

Conjectured bound $Y \geq 25$

If $Z < X + 25 \Rightarrow$ trouble!

- It is true that the more robust TLS infrastructure and experienced commmunity will be faster at deploying implementations.
- On the other hand:
  - NIST standardization analysis phase will take 5 years and 2 more for the drafts – D. Moody
  - The field of quantum cryptanalysis has only just begun (recent attacks against NTRU and binary MQ)
  - Two lines of attacks imply higher chances to break post-quantum assumptions.

Hash-based Signatures (HBS)

1976, Diffie-Hellman in "New directions in cryptography" introduce HBS:

One-way message authentication has a partial solution suggested to the authors by Leslie Lamport of Massachusetts Computer Associates. This technique employs a one-way function f mapping k-dimensional binary space into itself for k on the order of 100. If the transmitter wishes to send an N bit message he generates $2N$, randomly chosen, k-dimensional binary vectors $x_1, X_1, x_2, X_2, \cdots, x_N, X_N$ which he keeps secret. The receiver is given the corresponding images under f, namely $y_1, Y_1, y_2, Y_2, \cdots, y_N, Y_N$. Later, when the message $\boldsymbol{m} = (m_1, m_2, \cdots, m_N)$ is to be sent, the transmitter sends x_1 or X_1 depending on whether $m_1 = 0$ or 1. He sends x_2 or X_2 depending on whether $m_2 = 0$ or 1, etc. The receiver operates with f on the first received block and sees whether it yields y_1 or Y_1 as its image and thus learns whether it was x_1 or X_1, and whether $m_1 = 0$ or 1. In a similar manner the receiver is able to determine $m_2, m_3, \cdots, m_N$. But the receiver is incapable of forging a change in even one bit of $\boldsymbol{m}$.

1976, Lamport's signature: sign a n-bit message

Let $f : \{0,1\}^k \rightarrow \{0,1\}^k$ be a public one-way function

1976, Lamport's signature: sign a n-bit message

Let $f : \{0,1\}^k \to \{0,1\}^k$ be a public one-way function

Signer generates n pairs of k-bit strings $(x_i, X_i)_{i=1}^n \in_R \{0,1\}^k$:

$$sk = \{(x_1, X_1), (x_2, X_2), \cdots, (x_n, X_n)\}$$

1976, Lamport's signature: sign a n-bit message

Let $f : \{0,1\}^k \rightarrow \{0,1\}^k$ be a public one-way function

Signer generates n pairs of k-bit strings $(x_i, X_i)_{i=1}^n \in_R \{0,1\}^k$:

$$sk = \{(x_1, X_1), (x_2, X_2), \cdots, (x_n, X_n)\}$$

Evaluate f on them: $y_i, Y_i \leftarrow f(x_i), f(X_i)$ and publishes:

$$pk = \{(y_1, Y_1), (y_2, Y_2), \cdots, (y_n, Y_n)\}$$

1976, Lamport's signature: sign a n-bit message

Let $f : \{0,1\}^k \to \{0,1\}^k$ be a public one-way function

Signer generates n pairs of k-bit strings $(x_i, X_i)_{i=1}^n \in_R \{0,1\}^k$:

$$sk = \{(x_1, X_1), (x_2, X_2), \cdots, (x_n, X_n)\}$$

Evaluate f on them: $y_i, Y_i \leftarrow f(x_i), f(X_i)$ and publishes:

$$pk = \{(y_1, Y_1), (y_2, Y_2), \cdots, (y_n, Y_n)\}$$

To sign $m \in \{0,1\}^n$ reveal

$$\sigma_i \leftarrow \overline{m_i} x_i + m_i X_i \quad // m_i \text{ selects the } i\text{-th element}$$

1976, Lamport's signature: sign a n-bit message

Let $f : \{0,1\}^k \to \{0,1\}^k$ be a public one-way function

Signer generates n pairs of k-bit strings $(x_i, X_i)_{i=1}^n \in_R \{0,1\}^k$:

$$sk = \{(x_1, X_1), (x_2, X_2), \cdots, (x_n, X_n)\}$$

Evaluate f on them: $y_i, Y_i \leftarrow f(x_i), f(X_i)$ and publishes:

$$pk = \{(y_1, Y_1), (y_2, Y_2), \cdots, (y_n, Y_n)\}$$

To sign $m \in \{0,1\}^n$ reveal

$$\sigma_i \leftarrow \overline{m_i} x_i + m_i X_i \quad // m_i \text{ selects the } i\text{-th element}$$

To verify $\sigma = (\sigma_1, \ldots, \sigma_n)$ check if

$$f(\sigma_i) \stackrel{?}{=} \begin{cases} y_i \Rightarrow m_i = 0 \\ Y_i \Rightarrow m_i = 1 \end{cases}$$

Hash-based Signatures (HBS)

1976, Lamport's digital signature idea

- A problem pointed by Diffie and Hellman:

Hash-based Signatures (HBS)

1976, Lamport's digital signature idea

- A problem pointed by Diffie and Hellman:

  Imagine the message to be signed is $N = 1Mbit$.

Hash-based Signatures (HBS)

1976, Lamport's digital signature idea

- A problem pointed by Diffie and Hellman:

  Imagine the message to be signed is $N = 1Mbit$.

  The size of sk, pk would be $2kN \approx 2 \cdot 100 \cdot 10^6 = 200M$ bits

Hash-based Signatures (HBS)

1976, Lamport's digital signature idea

- A problem pointed by Diffie and Hellman:

  Imagine the message to be signed is $N = 1Mbit$.

  The size of sk, pk would be $2kN \approx 2 \cdot 100 \cdot 10^6 = 200M$ bits

- An improvement would be:

Hash-based Signatures (HBS)

1976, Lamport's digital signature idea

- A problem pointed by Diffie and Hellman:

  Imagine the message to be signed is $N = 1Mbit$.

  The size of sk, pk would be $2kN \approx 2 \cdot 100 \cdot 10^6 = 200M$ bits

- An improvement would be:

  Define a new one-way function $g : \{0,1\}^* \to \{0,1\}^n$.

Hash-based Signatures (HBS)

1976, Lamport's digital signature idea

- A problem pointed by Diffie and Hellman:

  Imagine the message to be signed is $N = 1Mbit$.

  The size of sk, pk would be $2kN \approx 2 \cdot 100 \cdot 10^6 = 200M$ bits

- An improvement would be:

  Define a new one-way function $g : \{0,1\}^* \to \{0,1\}^n$.

  Under message $m \in \{0,1\}^N$, compute $m' = g(m) \in \{0,1\}^n$

Hash-based Signatures (HBS)

1976, Lamport's digital signature idea

- A problem pointed by Diffie and Hellman:

  Imagine the message to be signed is $N = 1Mbit$.

  The size of sk, pk would be $2kN \approx 2 \cdot 100 \cdot 10^6 = 200M$ bits

- An improvement would be:

  Define a new one-way function $g : \{0,1\}^* \to \{0,1\}^n$.

  Under message $m \in \{0,1\}^N$, compute $m' = g(m) \in \{0,1\}^n$

  Generate sk and pk using $f : \{0,1\}^k \to \{0,1\}^k$ for each bit $1, \ldots, n$.

Hash-based Signatures (HBS)

1976, Lamport's digital signature idea

- A problem pointed by Diffie and Hellman:

  Imagine the message to be signed is $N = 1Mbit$.

  The size of sk, pk would be $2kN \approx 2 \cdot 100 \cdot 10^6 = 200M$ bits

- An improvement would be:

  Define a new one-way function $g : \{0,1\}^* \to \{0,1\}^n$.

  Under message $m \in \{0,1\}^N$, compute $m' = g(m) \in \{0,1\}^n$

  Generate sk and pk using $f : \{0,1\}^k \to \{0,1\}^k$ for each bit $1, \ldots, n$.

  Note that g can have many inputs mapped to a same output.

Hash-based Signatures (HBS)

1976, Lamport's digital signature idea

- A problem pointed by Diffie and Hellman:

  Imagine the message to be signed is $N = 1Mbit$.

  The size of sk, pk would be $2kN \approx 2 \cdot 100 \cdot 10^6 = 200M$ bits

- An improvement would be:

  Define a new one-way function $g : \{0,1\}^* \to \{0,1\}^n$.

  Under message $m \in \{0,1\}^N$, compute $m' = g(m) \in \{0,1\}^n$

  Generate sk and pk using $f : \{0,1\}^k \to \{0,1\}^k$ for each bit $1, \ldots, n$.

  Note that g can have many inputs mapped to a same output.

  Therefore, g should have stronger properties than f (collision resistance).

Hash-based Signatures (HBS)

Remark: Lamport-Diffie is a one-time signature (OTS)!

1. Assume a message $m_1 = (\mathbf{0}, 1, 1) \in \{0,1\}^3$ is signed. Thus

$$\begin{aligned}\sigma &= \{\overline{\mathbf{0}} x_1 + \mathbf{0} X_1, \overline{1} x_2 + 1 X_2, \overline{1} x_3 + 1 X_3\} \\ &= \{x_1, X_2, X_3\}\end{aligned}$$

Hash-based Signatures (HBS)

Remark: Lamport-Diffie is a one-time signature (OTS)!

1. Assume a message $m_1 = (\mathbf{0}, 1, 1) \in \{0,1\}^3$ is signed. Thus

$$\begin{aligned}\sigma &= \{\bar{\mathbf{0}} x_1 + \mathbf{0} X_1, \bar{1} x_2 + 1 X_2, \bar{1} x_3 + 1 X_3\} \\ &= \{x_1, X_2, X_3\}\end{aligned}$$

2. Let the signer produce a new signature of $m_2 = (1, 1, 0)$ under sk:

$$\sigma' = \{X_1, X_2, x_3\}$$

Hash-based Signatures (HBS)

Remark: Lamport-Diffie is a one-time signature (OTS)!

1. Assume a message $m_1 = (\mathbf{0}, 1, 1) \in \{0,1\}^3$ is signed. Thus

$$\sigma = \{\bar{\mathbf{0}}x_1 + \mathbf{0}X_1, \bar{1}x_2 + 1X_2, \bar{1}x_3 + 1X_3\}$$
$$= \{x_1, X_2, X_3\}$$

2. Let the signer produce a new signature of $m_2 = (1,1,0)$ under sk:

$$\sigma' = \{X_1, X_2, x_3\}$$

3. Notice that $\{x_1, X_1, X_2, x_3, X_3\}$ are now public.

Hash-based Signatures (HBS)

Remark: Lamport-Diffie is a one-time signature (OTS)!

1. Assume a message $m_1 = (\mathbf{0}, 1, 1) \in \{0,1\}^3$ is signed. Thus

$$\sigma = \{\bar{\mathbf{0}}x_1 + \mathbf{0}X_1, \bar{1}x_2 + 1X_2, \bar{1}x_3 + 1X_3\}$$
$$= \{x_1, X_2, X_3\}$$

2. Let the signer produce a new signature of $m_2 = (1,1,0)$ under sk:

$$\sigma' = \{X_1, X_2, x_3\}$$

3. Notice that $\{x_1, X_1, X_2, x_3, X_3\}$ are now public.
4. Then it's easy to forge a signature of $m_3 = (1,1,1)$ for example. Thus, Lamport-Diffie signature is OTS and each key pair can be only used once.

Hash-based Signatures (HBS)

Security assumption of Lamport-Diffie (LD)

1. one-way function f is hard to invert and
2. it is hard to find different input values that map to a same output

Put the above together, HBS rely on the existence of modern **cryptographically secure hash functions**.

Hash-based Signatures (HBS)

- Recall that LD's sk, pk keys consist of n pairs of k-bit strings.
- Can we do better in terms of space?

Hash-based Signatures (HBS)

- Recall that LD's *sk*, *pk* keys consist of n pairs of k-bit strings.
- Can we do better in terms of space?

Merkle 1979, an optimization of LD due to Winternitz

Idea: instead of processing $m \in \{0,1\}^n$ bit-by-bit, use w-bit chunks.

$$m = (m_1 || \cdots || m_{\lceil n/w \rceil})$$

where $m_i \in \{0,1\}^w$ can be viewed as w-bit integers.

Hash-based Signatures (HBS)

- Recall that LD's sk, pk keys consist of n pairs of k-bit strings.
- Can we do better in terms of space?

Merkle 1979, an optimization of LD due to Winternitz

Idea: instead of processing $m \in \{0,1\}^n$ bit-by-bit, use w-bit chunks.

$$m = (m_1 || \cdots || m_{\lceil n/w \rceil})$$

where $m_i \in \{0,1\}^w$ can be viewed as w-bit integers.

- KeyGen: Precompute $x_i, y_i = f^{2^w - 1}(x_i)$ for $i = 1, \cdots, \lceil n/w \rceil$ where $f^t(x) = f(\cdots f(x) \cdots)$ means t applications of f

Hash-based Signatures (HBS)

- Recall that LD's sk, pk keys consist of n pairs of k-bit strings.
- Can we do better in terms of space?

Merkle 1979, an optimization of LD due to Winternitz

Idea: instead of processing $m \in \{0,1\}^n$ bit-by-bit, use w-bit chunks.

$$m = (m_1 || \cdots || m_{\lceil n/w \rceil})$$

where $m_i \in \{0,1\}^w$ can be viewed as w-bit integers.

- KeyGen: Precompute $x_i, y_i = f^{2^w - 1}(x_i)$ for $i = 1, \cdots, \lceil n/w \rceil$ where $f^t(x) = f(\cdots f(x) \cdots)$ means t applications of f
- Actually, it is possible to do better:

$$pk = y = g(y_1 \parallel \cdots \parallel y_{\lceil n/w \rceil})$$

Hash-based Signatures (HBS)

- Recall that LD's sk, pk keys consist of n pairs of k-bit strings.
- Can we do better in terms of space?

Merkle 1979, an optimization of LD due to Winternitz

Idea: instead of processing $m \in \{0,1\}^n$ bit-by-bit, use w-bit chunks.

$$m = (m_1 || \cdots || m_{\lceil n/w \rceil})$$

where $m_i \in \{0,1\}^w$ can be viewed as w-bit integers.

- KeyGen: Precompute $x_i, y_i = f^{2^w - 1}(x_i)$ for $i = 1, \cdots, \lceil n/w \rceil$ where $f^t(x) = f(\cdots f(x) \cdots)$ means t applications of f
- Actually, it is possible to do better:

$$pk = y = g(y_1 \parallel \cdots \parallel y_{\lceil n/w \rceil})$$

- Public key pk boils down to one hash value (instead of $2n$).

Hash-based Signatures (HBS)

(cont. $\cdots$)

Winternitz OTS

- Sign: compute

$$\sigma = (f^{m_1}(x_1), \cdots, f^{m_{\lceil n/w \rceil}}(x_{\lceil n/w \rceil}))$$

Hash-based Signatures (HBS)

(cont. $\cdots$)

Winternitz OTS

- Sign: compute

$$\sigma = (f^{m_1}(x_1), \cdots, f^{m_{\lceil n/w \rceil}}(x_{\lceil n/w \rceil}))$$

- Verify: compute

$$y'_i = f^{2^w - 1 - m_i}(\sigma_i), \text{ for all } i$$
$$y' = g(y'_1 \parallel \cdots \parallel y'_{\lceil n/w \rceil})$$

Check

$$y' \stackrel{?}{=} y$$

Hash-based Signatures (HBS)

- **Problem**: Winternitz defined exactly as previously is **insecure**!

Hash-based Signatures (HBS)

- **Problem**: Winternitz defined exactly as previously is **insecure**!

  Assume a message $m = (m_1, \cdots, \mathbf{m_i}, \cdots, m_{\lceil n/w \rceil})$ with signature

$$\sigma = (f^{m_1}(x_1), \cdots, \mathbf{f^{m_i}}(\mathbf{x_i}), \cdots))$$

Hash-based Signatures (HBS)

- **Problem**: Winternitz defined exactly as previously is **insecure**!
  Assume a message $m = (m_1, \cdots, \mathbf{m_i}, \cdots, m_{\lceil n/w \rceil})$ with signature
  $$\sigma = (f^{m_1}(x_1), \cdots, \mathbf{f^{m_i}}(\mathbf{x_i}), \cdots))$$
- Assume that for some i we have $m_i < 2^w - 1$

Hash-based Signatures (HBS)

- **Problem**: Winternitz defined exactly as previously is **insecure**!
  Assume a message $m = (m_1, \cdots, \mathbf{m_i}, \cdots, m_{\lceil n/w \rceil})$ with signature

$$\sigma = (f^{m_1}(x_1), \cdots, \mathbf{f^{m_i}}(\mathbf{x_i}), \cdots))$$

- Assume that for some i we have $m_i < 2^w - 1$
- Then, an adversary can produce a valid signature for the message

$$m' = (m_1, \cdots, m_i + 1, \cdots, m_{\lceil n/w \rceil})$$

Hash-based Signatures (HBS)

- **Problem**: Winternitz defined exactly as previously is **insecure**!

  Assume a message $m = (m_1, \cdots, \mathbf{m_i}, \cdots, m_{\lceil n/w \rceil})$ with signature

$$\sigma = (f^{m_1}(x_1), \cdots, \mathbf{f^{m_i}(x_i)}, \cdots))$$

- Assume that for some i we have $m_i < 2^w - 1$
- Then, an adversary can produce a valid signature for the message

$$m' = (m_1, \cdots, m_i + 1, \cdots, m_{\lceil n/w \rceil})$$

  given by

$$\begin{aligned}\sigma' &= (f^{m_1}(x_1), \cdots, f(\mathbf{f^{m_i}(x_i)}), \cdots)) \\ &= (f^{m_1}(x_1), \cdots, \mathbf{f^{m_i+1}(x_i)}, \cdots)) \\ &= \sigma(m')\end{aligned}$$

Hash-based Signatures (HBS)

- **Problem**: Winternitz defined exactly as previously is **insecure**!

  Assume a message $m = (m_1, \cdots, \mathbf{m_i}, \cdots, m_{\lceil n/w \rceil})$ with signature

  $$\sigma = (f^{m_1}(x_1), \cdots, \mathbf{f^{m_i}(x_i)}, \cdots))$$

- Assume that for some i we have $m_i < 2^w - 1$
- Then, an adversary can produce a valid signature for the message

  $$m' = (m_1, \cdots, m_i + 1, \cdots, m_{\lceil n/w \rceil})$$

  given by

  $$\begin{aligned} \sigma' &= (f^{m_1}(x_1), \cdots, f(\mathbf{f^{m_i}(x_i)}), \cdots)) \\ &= (f^{m_1}(x_1), \cdots, \mathbf{f^{m_i+1}(x_i)}, \cdots)) \\ &= \sigma(m') \end{aligned}$$

- Violates the notion of existential unforgeability!

Hash-based Signatures (HBS)

- **Solution:** 1979, Merkle introduces a checksum.

Hash-based Signatures (HBS)

- **Solution:** 1979, Merkle introduces a checksum.
- From $m = (m_1, \cdots, m_{\lceil n/w \rceil})$, compute

$$CS = \sum_{i=1}^{\lceil n/w \rceil} 2^w - 1 - m_i$$

which can be stored in $t = log_2(\lceil n/w \rceil \cdot (2^w - 1))$ bits.

Hash-based Signatures (HBS)

- **Solution:** 1979, Merkle introduces a checksum.
- From $m = (m_1, \cdots, m_{\lceil n/w \rceil})$, compute

$$CS = \sum_{i=1}^{\lceil n/w \rceil} 2^w - 1 - m_i$$

which can be stored in $t = log_2(\lceil n/w \rceil \cdot (2^w - 1))$ bits.

- **Idea**: represent CS as w-bit chunks as well:

$$CS = (m_{\lceil n/w \rceil + 1}, \cdots, m_{\lceil n/w \rceil + \lceil t/w \rceil})$$

Hash-based Signatures (HBS)

- **Solution:** 1979, Merkle introduces a checksum.
- From $m = (m_1, \cdots, m_{\lceil n/w \rceil})$, compute

$$CS = \sum_{i=1}^{\lceil n/w \rceil} 2^w - 1 - m_i$$

  which can be stored in $t = log_2(\lceil n/w \rceil \cdot (2^w - 1))$ bits.
- **Idea**: represent CS as w-bit chunks as well:

$$CS = (m_{\lceil n/w \rceil + 1}, \cdots, m_{\lceil n/w \rceil + \lceil t/w \rceil})$$

- Extend the message to be $m || CS = (m_1, \cdots, m_{\lceil n/w \rceil + \lceil t/w \rceil})$

Hash-based Signatures (HBS)

(checksum: cont ...)

- Given a signature

$$\sigma = (f^{m_1}(x_1), \cdots, \mathbf{f^{m_i}(x_i)}, \cdots))$$

Hash-based Signatures (HBS)

(checksum: cont ...)

- Given a signature

$$\sigma = (f^{m_1}(x_1), \cdots, \mathbf{f}^{\mathbf{m_i}}(\mathbf{x_i}), \cdots))$$

- Let $i \leq \lceil n/w \rceil$. If the adversary tries to go forward on $\mathbf{f}^{\mathbf{m_i}}(\mathbf{x_i})$:

$$\begin{aligned} \sigma' &= (f^{m_1}(x_1), \cdots, f(\mathbf{f}^{\mathbf{m_i}}(\mathbf{x_i})), \cdots)) \\ &= (f^{m_1}(x_1), \cdots, \mathbf{f}^{\mathbf{m_i+1}}(\mathbf{x_i}), \cdots)) \end{aligned}$$

Hash-based Signatures (HBS)

(checksum: cont ...)

- Given a signature

$$\sigma = (f^{m_1}(x_1), \cdots, \mathbf{f^{m_i}(x_i)}, \cdots))$$

- Let $i \leq \lceil n/w \rceil$. If the adversary tries to go forward on $\mathbf{f^{m_i}(x_i)}$:

$$\begin{aligned}\sigma' &= (f^{m_1}(x_1), \cdots, f(\mathbf{f^{m_i}(x_i)}), \cdots)) \\ &= (f^{m_1}(x_1), \cdots, \mathbf{f^{m_i+1}(x_i)}, \cdots))\end{aligned}$$

- Then the new checksum is $CS' = CS - 1$ which implies an inversion $f^{-1}(f^{m_j}(x_j))$ for some $m_j \in CS$.

Hash-based Signatures (HBS)

(checksum: cont ...)

- Given a signature

$$\sigma = (f^{m_1}(x_1), \cdots, \mathbf{f^{m_i}(x_i)}, \cdots))$$

- Let $i \leq \lceil n/w \rceil$. If the adversary tries to go forward on $\mathbf{f^{m_i}(x_i)}$:

$$\begin{aligned}\sigma' &= (f^{m_1}(x_1), \cdots, f(\mathbf{f^{m_i}(x_i)}), \cdots)) \\ &= (f^{m_1}(x_1), \cdots, \mathbf{f^{m_i+1}(x_i)}, \cdots))\end{aligned}$$

- Then the new checksum is $CS' = CS - 1$ which implies an inversion $f^{-1}(f^{m_j}(x_j))$ for some $m_j \in CS$.
- If a forger targets $m_j \in CS$, an inversion is also implied on m_i. Thus, the checksum protects against such attacks.

Winternitz OTS example

- Let $w = 2$ and one wants to sign the $(n = 4)$-bit message

$$m = (1011)$$

with $\lceil n/w \rceil = 2$ and $m = (m_1 = 2, m_2 = 3)$.

Winternitz OTS example

- Let $w = 2$ and one wants to sign the ($n = 4$)-bit message

$$m = (1011)$$

  with $\lceil n/w \rceil = 2$ and $m = (m_1 = 2, m_2 = 3)$.
- Compute the checksum:

$$\begin{aligned} CS &= \sum_{i=1}^{2} 2^2 - 1 - m_i \\ &= 3 - 2 + 3 - 3 = 1 \end{aligned}$$

Winternitz OTS example

- Let $w = 2$ and one wants to sign the $(n = 4)$-bit message

$$m = (1011)$$

  with $\lceil n/w \rceil = 2$ and $m = (m_1 = 2, m_2 = 3)$.

- Compute the checksum:

$$\begin{aligned} CS &= \sum_{i=1}^{2} 2^2 - 1 - m_i \\ &= 3 - 2 + 3 - 3 = 1 \end{aligned}$$

  Thus, $CS = (0001) = (m_3 = 0, m_4 = 1)$ and the actual message is

$$m || CS = (2, 3, 0, 1)$$

(*cont* ...) Winternitz OTS: example

- Key generation including CS:

$$sk = (x_1, \cdots, x_4)$$
$$pk = y = g(f^3(x_1) \parallel \cdots \parallel f^3(x_4))$$

(*cont . . .*) Winternitz OTS: example

- Key generation including CS:

$$sk = (x_1, \cdots, x_4)$$
$$pk = y = g(f^3(x_1) \parallel \cdots \parallel f^3(x_4))$$

- The signature of $m||CS = (2, 3, 0, 1)$ will be

$$\sigma = (f^2(x_1), f^3(x_2), x_3, f(x_4)) = (\sigma_1, \sigma_2, \sigma_3, \sigma_4)$$

(*cont ...*) Winternitz OTS: example

- Key generation including CS:

$$sk = (x_1, \cdots, x_4)$$
$$pk = y = g(f^3(x_1) \parallel \cdots \parallel f^3(x_4))$$

- The signature of $m||CS = (2, 3, 0, 1)$ will be

$$\sigma = (f^2(x_1), f^3(x_2), x_3, f(x_4)) = (\sigma_1, \sigma_2, \sigma_3, \sigma_4)$$

- Verification of σ will be

(*cont . . .*) Winternitz OTS: example

- Key generation including CS:

$$sk = (x_1, \cdots, x_4)$$
$$pk = y = g(f^3(x_1) \parallel \cdots \parallel f^3(x_4))$$

- The signature of $m||CS = (2, 3, 0, 1)$ will be

$$\sigma = (f^2(x_1), f^3(x_2), x_3, f(x_4)) = (\sigma_1, \sigma_2, \sigma_3, \sigma_4)$$

- Verification of σ will be
  - Recompute CS from m getting $m||CS = (2, 3, 0, 1)$

(*cont ...*) Winternitz OTS: example

- Key generation including CS:

$$sk = (x_1, \cdots, x_4)$$
$$pk = y = g(f^3(x_1) \parallel \cdots \parallel f^3(x_4))$$

- The signature of $m||CS = (2, 3, 0, 1)$ will be

$$\sigma = (f^2(x_1), f^3(x_2), x_3, f(x_4)) = (\sigma_1, \sigma_2, \sigma_3, \sigma_4)$$

- Verification of σ will be
  - Recompute CS from m getting $m||CS = (2, 3, 0, 1)$
  - Compute and check

$$(y_1', y_2', y_3', y_4') = (f^{3-\mathbf{2}}(\sigma_1), f^{3-\mathbf{3}}(\sigma_2), f^{3-\mathbf{0}}(\sigma_3), f^{3-\mathbf{1}}(\sigma_4))$$

$$g(y_1' \parallel y_2' \parallel y_3' \parallel y_4') \stackrel{?}{=} y$$

Hash-based Signatures (HBS)

Note on the efficiency of Winternitz OTS

- $|sk| = |\sigma| \approx (\mathbf{n}/\mathbf{w})\mathbf{k}$ (ignoring the checksum)

Hash-based Signatures (HBS)

Note on the efficiency of Winternitz OTS

- $|sk| = |\sigma| \approx (\mathbf{n}/\mathbf{w})\mathbf{k}$ (ignoring the checksum)
  - Compares well with the **2nk** bits in $|sk|$, **nk** in $|\sigma|$ for LD
  - A **2w** reduction factor for sk
  - A **w** reduction factor for σ

Hash-based Signatures (HBS)

Note on the efficiency of Winternitz OTS

- $|sk| = |\sigma| \approx (\mathbf{n}/\mathbf{w})\mathbf{k}$ (ignoring the checksum)
  - Compares well with the **2nk** bits in $|sk|$, **nk** in $|\sigma|$ for LD
  - A **2w** reduction factor for sk
  - A **w** reduction factor for σ
- $|pk| = \mathbf{n}$ (vs **2nk** in LD)

Hash-based Signatures (HBS)

Note on the efficiency of Winternitz OTS

- $|sk| = |\sigma| \approx (\mathbf{n}/\mathbf{w})\mathbf{k}$ (ignoring the checksum)
  - Compares well with the **2nk** bits in $|sk|$, **nk** in $|\sigma|$ for LD
  - A **2w** reduction factor for *sk*
  - A **w** reduction factor for σ
- $|pk| = \mathbf{n}$ (vs **2nk** in LD)
- But, WOTS requires $(\mathbf{2^w} - \mathbf{1})/\mathbf{w}$ hash evaluations per bit
  - While LD requires **2** evaluations per bit
  - Notice that for $w = 1$ we get exactly Lamport-Diffie

Hash-based Signatures (HBS)

Note on the efficiency of Winternitz OTS

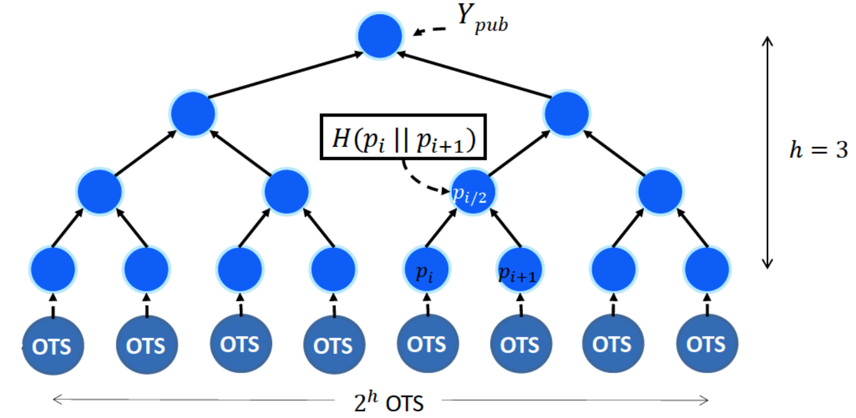
- $|sk| = |\sigma| \approx (\mathbf{n}/\mathbf{w})\mathbf{k}$ (ignoring the checksum)
  - Compares well with the $\mathbf{2nk}$ bits in $|sk|$, $\mathbf{nk}$ in $|\sigma|$ for LD
  - A $\mathbf{2w}$ reduction factor for sk
  - A $\mathbf{w}$ reduction factor for σ
- $|pk| = \mathbf{n}$ (vs $\mathbf{2nk}$ in LD)
- But, WOTS requires $(\mathbf{2^w} - \mathbf{1})/\mathbf{w}$ hash evaluations per bit
  - While LD requires $\mathbf{2}$ evaluations per bit
  - Notice that for $w = 1$ we get exactly Lamport-Diffie
- Since hash evaluations can be very fast, it is a reasonable tradeoff

Hash-based Signatures (HBS)

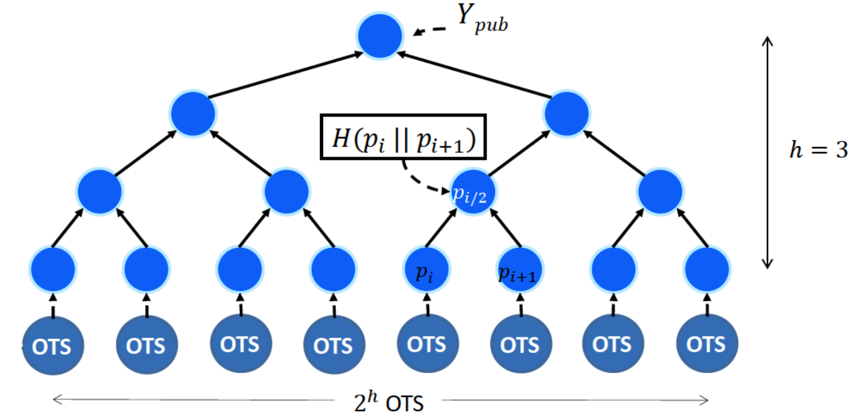
Scheme	$n = k$	$\lvert PrivKey \rvert$	$\lvert PubKey \rvert$	$\lvert Sig \rvert$
LD OTS	256	16	16	16
WOTS (w=2)	256	4.2	32 bytes	4.2
WOTS (w=8)	256	1.1	32 bytes	1.1
WOTS (w=16)	256	0.6	32 bytes	0.6

Table: Parameter sizes for one-time signatures in KiB

1979, Merkle turns OTS into multi-time signatures



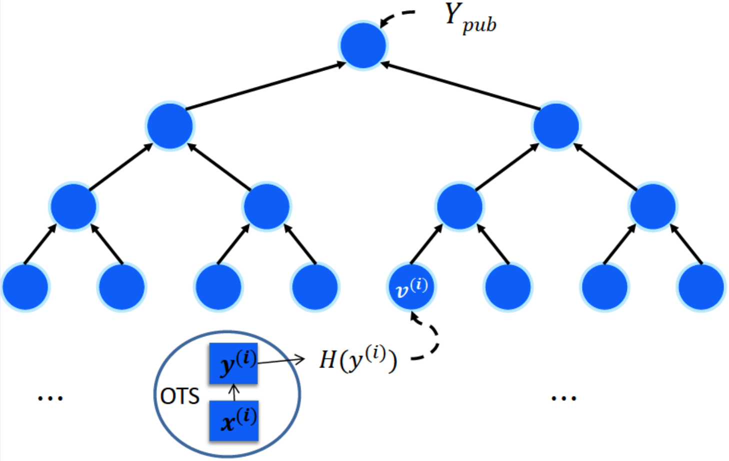
1979, Merkle turns OTS into multi-time signatures



- OTS can be **any** one-time signature scheme.
- Y_{pub} authenticates 2^h OTS key pairs.

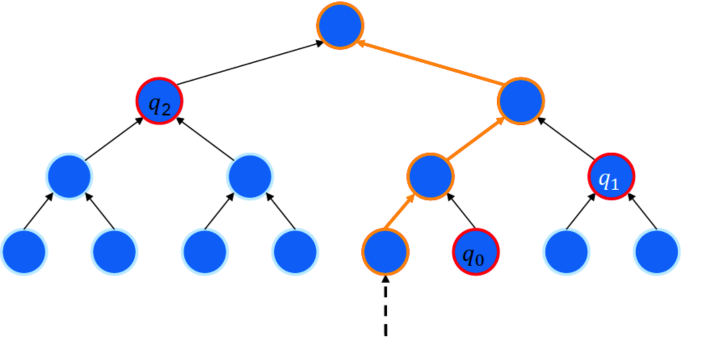
Hash-based Signatures (HBS)

1979, Merkle signature: **Key Generation**



Hash-based Signatures (HBS)

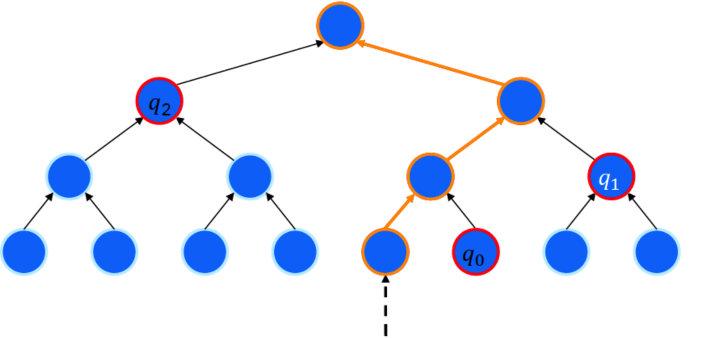
1979, Merkle signature: **Sign**



$$\sigma(data) = \left(i, Sign_{OTS}\left(x^{(i)}, data\right), v^{(i)}, (q_0, \ldots, q_{h-1})\right)$$

Hash-based Signatures (HBS)

1979, Merkle signature: **Sign**

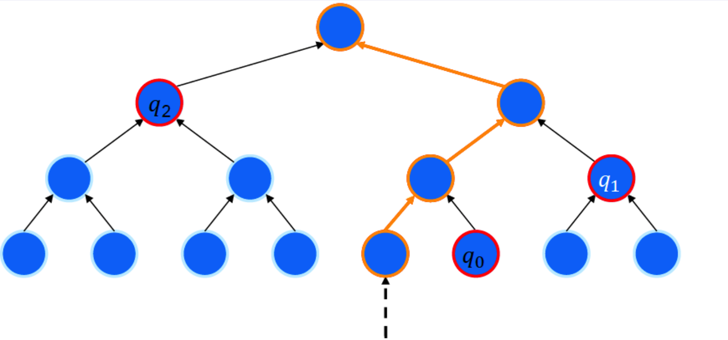


$$\sigma(data) = \left(i, Sign_{OTS}\left(x^{(i)}, data\right), v^{(i)}, (q_0, \ldots, q_{h-1})\right)$$

- Nodes q_i are called the **authentication path** of i-th signature

Hash-based Signatures (HBS)

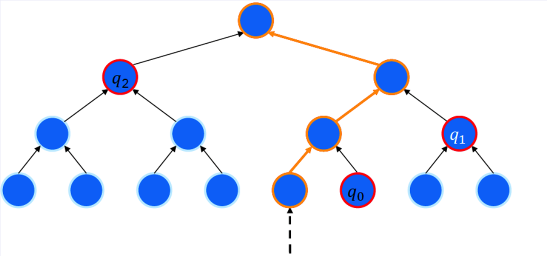
1979, Merkle signature: **Sign**



$$\sigma(data) = \left(i, Sign_{OTS}\left(x^{(i)}, data\right), v^{(i)}, (q_0, \dots, q_{h-1})\right)$$

- Nodes q_i are called the **authentication path** of i-th signature
- Stateful: susceptible to some attacks, e.g. 'restart attacks'

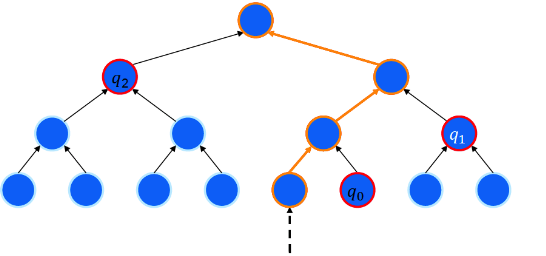
Time efficiency of the Merkle signature



$$\sigma(data) = \left(i, Sign_{OTS}\left(x^{(i)}, data\right), v^{(i)}, (q_0, \ldots, q_{h-1})\right)$$

- Requires $O(2^h)$ hash evaluations per signature

Time efficiency of the Merkle signature

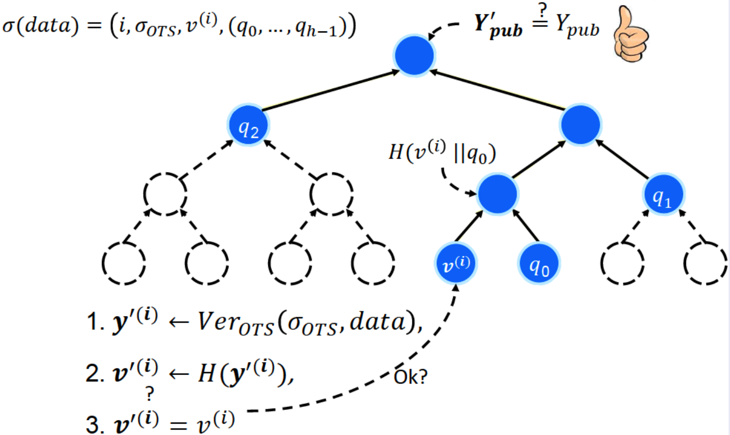


$$\sigma(data) = \big(i, Sign_{OTS}(x^{(i)}, data), v^{(i)}, (q_0, \dots, q_{h-1})\big)$$

- Requires $O(2^h)$ hash evaluations per signature
- Improvement by BDS'08.
  - Store strategic (higher) nodes on a state during KeyGen.
  - Allows for a tradeoff between size of the state *vs* # leaf computations at each signature.

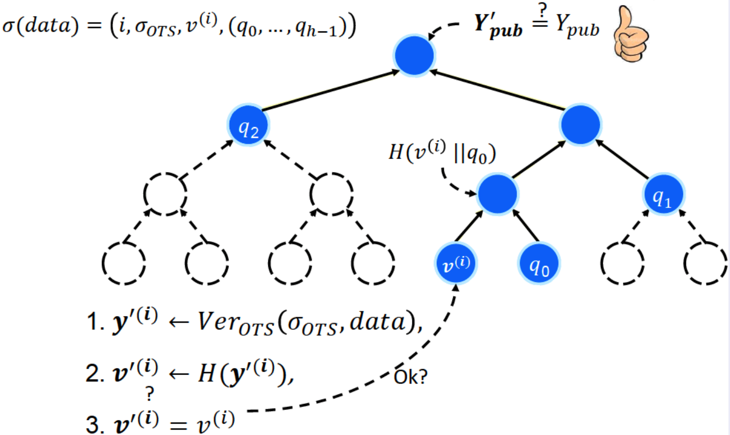
Hash-based Signatures (HBS)

1979, Merkle signature: **Verify**



Hash-based Signatures (HBS)

1979, Merkle signature: **Verify**



- An obvious optimization is not sending $v^{(i)}$. Verifier only checks the root.

Space efficiency of the Merkle signature

- Private key size: $2^h \cdot |sk_{OTS}|$
- Public key size: size of hash H, e.g. 256 bits.
- Signature size: $|\sigma| = |i| + |\sigma_{OTS}| + |v(i)| + |(q_0, \cdots, q_{h-1})|$

Space efficiency of the Merkle signature

- Private key size: $2^h \cdot |sk_{OTS}|$
- Public key size: size of hash H, e.g. 256 bits.
- Signature size: $|\sigma| = |i| + |\sigma_{OTS}| + |v(i)| + |(q_0, \cdots, q_{h-1})|$

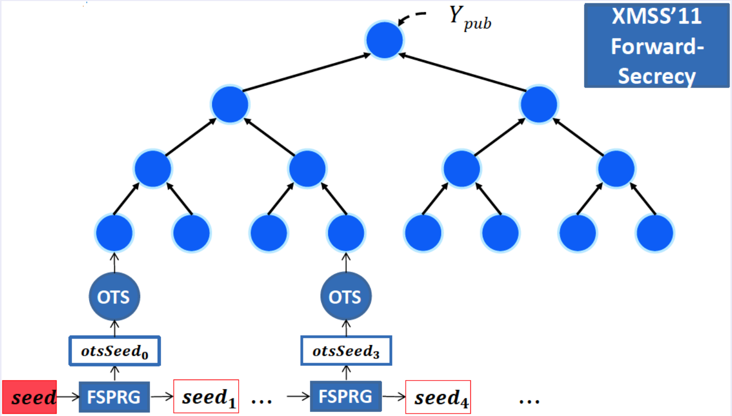
Merkle parameter sizes example

- $|f| = |H| = n = 256, h = 10$

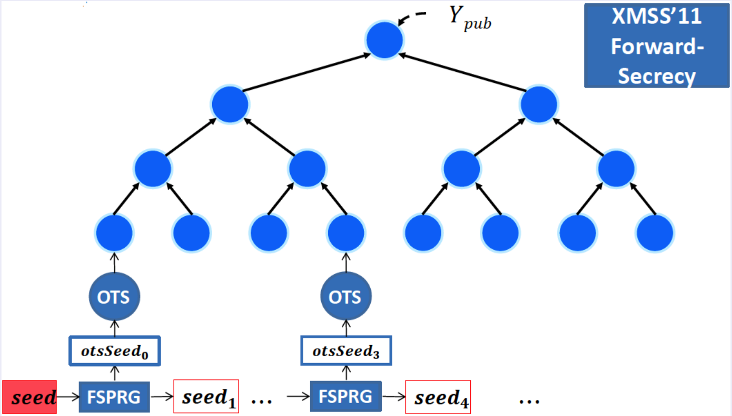
Scheme	$\lvert PrivKey \rvert$	$\lvert PubKey \rvert$	$\lvert Sig \rvert$
Merkle+LD	16 MiB	32 bytes	16.4 KiB
Merkle+WOTS (w=2)	4.2 MiB	32 bytes	4.5 KiB
Merkle+WOTS (w=16)	0.6 MiB	32 bytes	0.9 KiB

Table: Parameter sizes for Merkle multi-time signature (**1024 signatures**)

Merkle signature: XMSS'11 introduces additional properties

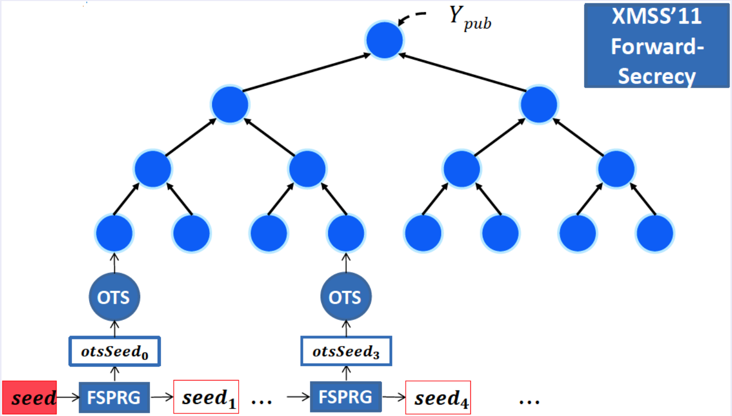


Merkle signature: XMSS'11 introduces additional properties



- XMSS uses the variant WOTS$^+$. Collision-resistance unecessary.

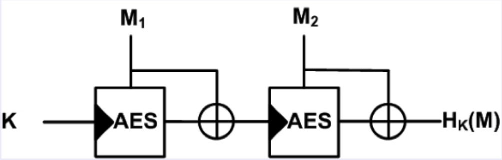
Merkle signature: XMSS'11 introduces additional properties



- XMSS uses the variant WOTS$^+$. Collision-resistance unecessary.
- Implication: half-size hashes can be used safely.

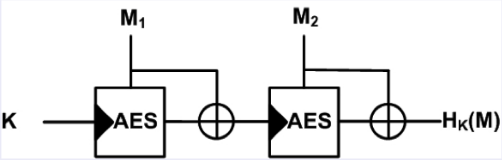
Merkle signature: implemention of PRNG and hash function

- Matyas-Meyer-Oseas: block-cipher-based hash function



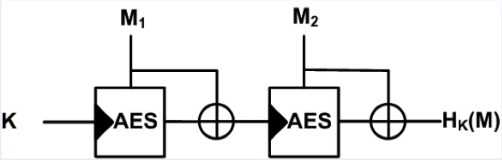
Merkle signature: implemention of PRNG and hash function

- Matyas-Meyer-Oseas: block-cipher-based hash function

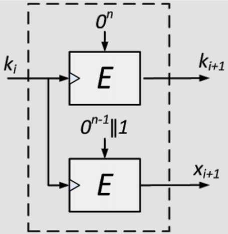


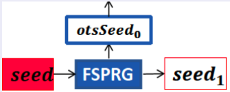
- Fast optimized (hw/sw) block-ciphers available in many platforms

- Matyas-Meyer-Oseas: block-cipher-based hash function



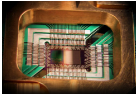
- Fast optimized (hw/sw) block-ciphers available in many platforms
- FSPRG by Standaert et al. 2010:



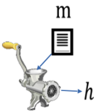


Hash-based Signatures (HBS) – A holistic view

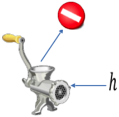
Post-quantum security



Only require hash functions
(efficient/minimal security assumption)



No reliance on trapdoors



Robust security (1976)
(cryptanalysis with little progress)

www.ieee.org/about/awards/bios/hamming_recipients

Larger signatures

References I

Shor, Peter W (1994).
"Algorithms for quantum computation: Discrete logarithms and factoring".

Proos, John and Christof Zalka (2003).
"Shor's discrete logarithm quantum algorithm for elliptic curves".

Häner, Thomas, Martin Roetteler, and Krysta M Svore (2016).
"Factoring using 2n+2 qubits with Toffoli based modular multiplication".

Roetteler, Martin et al. (2017).
"Quantum resource estimates for computing elliptic curve discrete logarithms".

Grover, Lov K (1996).
"A fast quantum mechanical algorithm for database search".

Bennett, Charles H et al. (1997).
"Strengths and weaknesses of quantum computing".

Merkle, Ralph Charles (1979).
"Secrecy, authentication, and public key systems".

References II

Standaert, FranCcois-Xavier et al. (2010).
"Leakage resilient cryptography in practice".