



Introduction to Post-Quantum Cryptography. Hash-Based Signatures.

Applied Cryptography – Spring 2024

Simona Samardjiska

May 22, 2024

Institute for Computing and Information Sciences
Radboud University

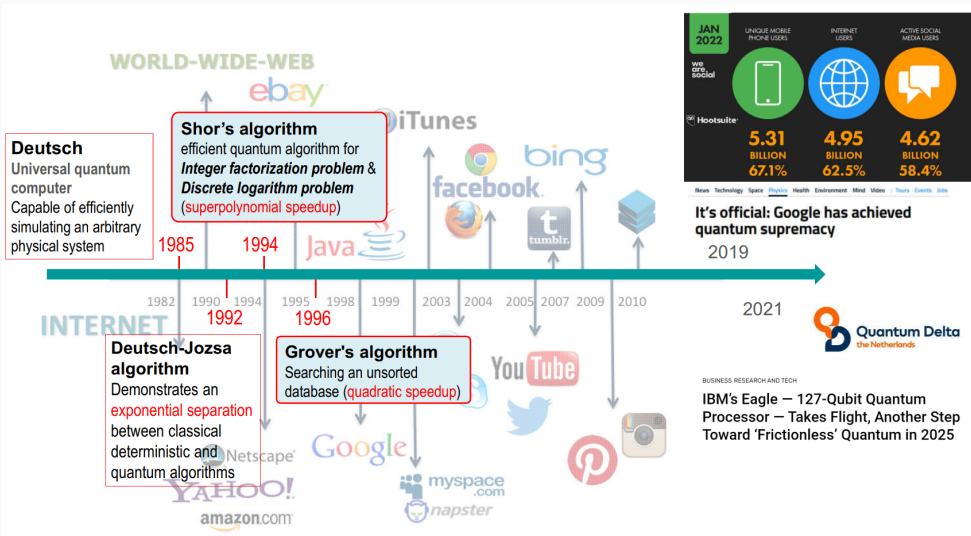
Last time in public key crypto:

- Commitment schemes
- Zero-Knowledge protocols
- Identification Protocols

Today:

- Identification Protocols ctd.
- Fiat-Shamir Signatures
- DSA and ECDSA
- Hash-Based Signatures (in this slide set)

A timeline of developments...



What can an adversary in a possession of a quantum computer do?

Using Shor's algorithm '94

Polynomial algorithm ($O(n^3)$) for

- Integer factorization
- Discrete logarithm problem

- **Break RSA**
- **Break DSA**
- **Break ECDSA**
- **Break Diffie-Hellman key exchange**
- ...

To factor a **2048 bit** number:

Classically ~ 150,000 years

Using Shor < 1 second



Today's cryptography?

**Broken
or
needs serious revision!**

Using Grover's algorithm '96

Polynomial algorithm ($O(\sqrt{N})$) for searching an unsorted database

Quadratic speedup

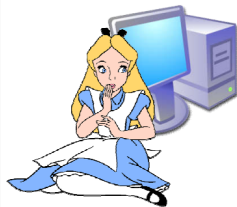
- password cracking
- Key searching (for ex. AES, MACs)
- Preimage finding (hash functions)
- Solutions for NP hard problems
- ...

To crack a **8 character** password of only lowercase letters

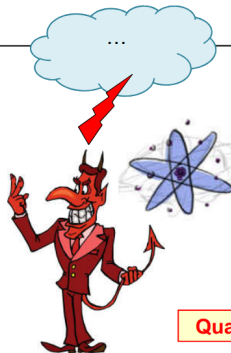
Classically ~ 4,13 years

Using Grover < 5 days

Classical cryptosystems believed to be secure
against quantum computer attacks



Classical



Quantum



Classical

Isogeny based cryptosystems –

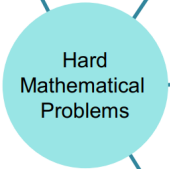
KEMs/NIKEs/signatures

(Finding isogenies on supersingular elliptic curves)

Multivariate Quadratic cryptosystems – mainly signatures

(Polynomial System Solving –PoSSo, for quadratic polynomials - MQ problem)

Hard
Mathematical
Problems



Code-based cryptosystems – mainly encryption/KEMs

(decoding random linear codes)

Hash-based signatures (only)

(only secure hash function needed)

Lattice-based cryptosystems – signatures/encryption/KEMs

(many different hard problems – SIS, SVP, LWE)

Hash-Based Signatures

Lamport One-Time Signature (OTS) - '79

- Very interesting construction, that happens to use **only a hash function**
- **No other mathematical hard problem necessary!**

The construction:

Let $H : \{0, 1\}^* \rightarrow \{0, 1\}^*$ be a function. Construct a signature scheme for messages of length $\ell = \ell(n)$ as follows:

- **Gen:** on input 1^n , proceed as follows for $i \in \{1, \dots, \ell\}$:
 1. Choose uniform $x_{i,0}, x_{i,1} \in \{0, 1\}^n$.
 2. Compute $y_{i,0} := H(x_{i,0})$ and $y_{i,1} := H(x_{i,1})$.

The public key pk and the private key sk are

$$pk = \begin{pmatrix} y_{1,0} & y_{2,0} & \cdots & y_{\ell,0} \\ y_{1,1} & y_{2,1} & \cdots & y_{\ell,1} \end{pmatrix} \quad sk = \begin{pmatrix} x_{1,0} & x_{2,0} & \cdots & x_{\ell,0} \\ x_{1,1} & x_{2,1} & \cdots & x_{\ell,1} \end{pmatrix}.$$

Lamport One-Time Signature (OTS) - '79

The construction (contd.):

- **Sign:** on input a private key sk as above and a message $m \in \{0, 1\}^\ell$ with $m = m_1 \cdots m_\ell$, output the signature $(x_{1,m_1}, \dots, x_{\ell,m_\ell})$.
- **Vrfy:** on input a public key pk as above, a message $m \in \{0, 1\}^\ell$ with $m = m_1 \cdots m_\ell$, and a signature $\sigma = (x_1, \dots, x_\ell)$, output 1 if and only if $H(x_i) = y_{i,m_i}$ for all $1 \leq i \leq \ell$.

Example:

Signing $m = 011$:

$$sk = \begin{pmatrix} \boxed{x_{1,0}} & x_{2,0} & x_{3,0} \\ x_{1,1} & \boxed{x_{2,1}} & \boxed{x_{3,1}} \end{pmatrix} \Rightarrow \sigma = (x_{1,0}, x_{2,1}, x_{3,1})$$

Verifying for $m = 011$ and $\sigma = (x_1, x_2, x_3)$:

$$pk = \left(\begin{pmatrix} \boxed{y_{1,0}} & y_{2,0} & y_{3,0} \\ y_{1,1} & \boxed{y_{2,1}} & \boxed{y_{3,1}} \end{pmatrix} \right) \Rightarrow \begin{aligned} H(x_1) &\stackrel{?}{=} y_{1,0} \\ H(x_2) &\stackrel{?}{=} y_{2,1} \\ H(x_3) &\stackrel{?}{=} y_{3,1} \end{aligned}$$

Lamport One-Time Signature (OTS) - '79

- Works **only for one** signature!
- Trivially forgeable given two valid signatures (if different in more than one bit)

Example:

$$\begin{pmatrix} \boxed{x_{1,0}} & x_{2,0} & x_{3,0} \\ x_{1,1} & \boxed{x_{2,1}} & \boxed{x_{3,1}} \end{pmatrix} \Rightarrow \sigma = (x_{1,0}, x_{2,1}, x_{3,1})$$

$$\begin{pmatrix} x_{1,0} & \boxed{x_{2,0}} & x_{3,0} \\ \boxed{x_{1,1}} & x_{2,1} & \boxed{x_{3,1}} \end{pmatrix} \Rightarrow \sigma = (x_{1,1}, x_{2,0}, x_{3,1})$$

“Mix and match” forgery

$$\begin{pmatrix} \boxed{x_{1,0}} & \boxed{x_{2,0}} & x_{3,0} \\ x_{1,1} & x_{2,1} & \boxed{x_{3,1}} \end{pmatrix} \Rightarrow \sigma = (x_{1,0}, x_{2,0}, x_{3,1})$$

Security of One-Time Signatures

Existential unforgeability under **single message attack (EUF-SMA)**

A Digital Signature scheme D_{ss} is called EUF-SMA-secure (one-time-EUF-CMA-secure) if any PPT algorithm \mathcal{A} has only negligible success probability

$$\text{Succ}_{D_{ss}(1^k)}^{\text{euf-sma}}(\mathcal{A}) = \Pr \left[\text{Exp}_{D_{ss}(1^k)}^{\text{euf-sma}}(\mathcal{A}) = \text{Accept} \right].$$

in the following $\text{Exp}_{D_{ss}(1^k)}^{\text{euf-sma}}(\mathcal{A})$ **game (experiment)**:

Challenger

Adversary

$(pk, sk) \leftarrow \text{KGen}()$

pk

M'

$\text{Sign}(sk, M')$

(M^*, σ^*)

single message M'

M^*, σ^*

Return 1 iff $\forall f(pk, M, \sigma) = \text{Accept}$ otherwise 0.

If H is one-way function, Lamport OTS is one-time secure.

Idea of proof:

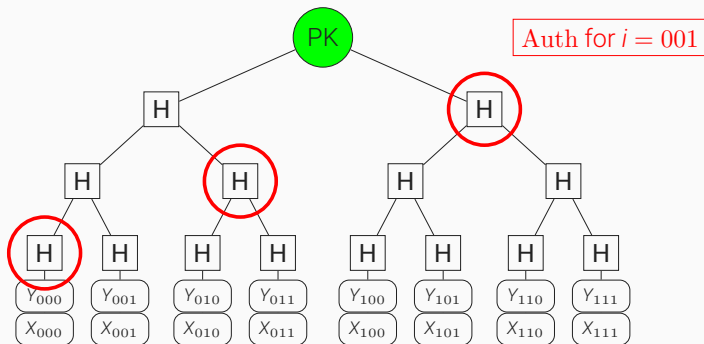
- During the game, \mathcal{A} asks for the signature of M'
- Consider any other message $M \neq M'$
- There must be at least one position $i \in \{1, \dots, \ell\}$ on which they differ. Let $M_i = b \neq M'_i$.
- Then forging a signature on M requires, at least, finding a preimage (under H) of element $y_{i,b}$ of the public key.
- Since H is one-way, this is infeasible.

Technical hints:

- Inverter \mathcal{I} wants to invert H at given y
- Prepares “appropriate” input public key to \mathcal{A} , s.t. $y_{i^*, b^*} = y$ for a randomly chosen i^*, b^*
- The rest of the public key is generated properly, as in the key generation
- Probability of success - $1/2\ell$ times probability of success of \mathcal{A}

Merkle Signature Scheme (MSS) - '89

- Merkle extends Lamport's OTS to multi-time signature scheme, using binary tree (Merkle tree)
- Each leaf is public key of OTS
- OTS are used sequentially (must keep track of index! - **stateful** signature scheme)
- Signature is $\sigma = (i, \text{Sign}_{\text{OTS}}(M, X_i), Y_i, \text{Auth})$



Merkle Signature Scheme (MSS) - '89

- Let $H : \{0, 1\}^* \rightarrow \{0, 1\}^s$ be a cryptographic hash function
- Assume that a one-time signature scheme (OTS) is given
- For $h \in \mathbb{N}$, 2^h signatures are to be generated that are verifiable with one MSS public key.

MSS:

KeyGen:

- ① generate 2^h OTSS key pairs (X_i, Y_i) , $i \in \{1, \dots, 2^h\}$, where the X^i are the signature keys and Y^i the verification keys.
- ② The MSS private key is (X_1, \dots, X_{2^h}) .
- ③ To determine the MSS public key, construct a binary authentication tree as follows. The leafs of the authentication tree are the hash values $H(Y_i)$ of the verification keys. Each inner node (including the root) of the tree is the hash value of the concatenation of its two children. The MSS public key is the root of the authentication tree.

MSS:

Sign: Given message M ,

- ① The i -th message is signed with the i -th key par X_i, Y_i .
- ② The signature is $\sigma = (i, \text{Sign}_{OTS}(M, X_i), Y_i, \text{Auth})$
- ③ The authentication path Auth is a sequence of nodes (a_h, \dots, a_1) in the authentication tree - the siblings needed to go up in the tree

Verify: To verify the message - signature pair (M, σ)

- ① The verifier first verifies the one-time signature with the verification key Y_i . If this verification fails, the verifier rejects the MSS signature as invalid.
- ② Otherwise, the verifier constructs a sequence of nodes of the tree of length $h + 1$ starting from the hash $H(Y_i)$ of the verification key Y_i .
 - Each node is constructed as the hash of the previous node and its sibling that is part of the authentication path
- ③ The verifier accepts if the last node in the sequence is the MSS public key.

MSS is **EUF-CMA secure** if OTS is a one-time EUF-CMA secure signature scheme and H is a random element from a family of collision resistant hash functions.

- Suppose an adversary \mathcal{B} exists that breaks the EUF-CMA security of MSS.
- We build an adversary \mathcal{A} that forges an OTS or finds a collision for H
- Procedure:
 - ① In the one-time EUF-CMA security game, \mathcal{A} receives a public key Y , which it puts as leaf i_* . It also has access to the one-time signing oracle corresponding to this public key
 - ② \mathcal{A} calls the KGen algorithm of OTS, to obtain 2^N pairs $(X_i, Y_i), i \neq i_*$ of private-public keys
 - ③ It constructs a Merkle tree of depth N , with root R
 - ④ It runs \mathcal{B} against the Merkle signature
 - ⑤ In the game, \mathcal{B} asks the signatures of various messages and \mathcal{A} computes genuine signatures σ_i for all queried messages. If \mathcal{B} asks for a signature for i^* , \mathcal{A} uses the signing oracle it has access to, to get a valid signature

- Procedure (contd.):
 - ⑥ \mathcal{B} outputs a forgery $(M', \sigma' = (j^*, \text{Sign}_{OTS}(M', X'_{j^*}), Y'_{j^*}, \text{Auth}'))$
 - ⑦ After the forgery is output, \mathcal{A} checks whether the number of queries is higher than j^* . If yes, a message was queried and a genuine signature has been produced for j^* . If not, it continues to sign the message until it reaches the index j^* .
 - ⑧ Hence, there exists a genuine signature for j^* : $(M'', \sigma = (j^*, \text{Sign}_{OTS}(M'', X_{j^*}), Y_{j^*}, \text{Auth}))$
 - ⑨ If $(Y_{j^*}, \text{Auth}) = (Y'_{j^*}, \text{Auth}')$, then the adversary \mathcal{B} used the same public key as \mathcal{A} , and thus we have a forgery for the OTS (can be used only if $j^* = i^*$, because we need a forgery for the i^* -th public key)
 - ⑩ If $(Y_{j^*}, \text{Auth}) \neq (Y'_{j^*}, \text{Auth}')$, the auth. paths are different, but the root is the same. Hence, somewhere along the path there is a collision.
- Success probability:
 - Let ϵ be the success probability of the forger \mathcal{B}
 - If ρ - probability that genuine and forged auth. paths are different, collision can be found with prob. $\rho\epsilon$,
 - Otherwise success of $\frac{1}{2^N} \cdot (1 - \rho)\epsilon$ to have $i = i^*$

- Lamport's scheme produces extremely large signatures
 - for ex. for hash function with 256-bit output, in order to sign 256-bit messages, the keys are of size $2 \times 256 \times 32B = 16KB$ and the signature $256 \times 32B = 8KB$
- Idea - **hash chains**
- Winternitz OTS

- **Key generation:**

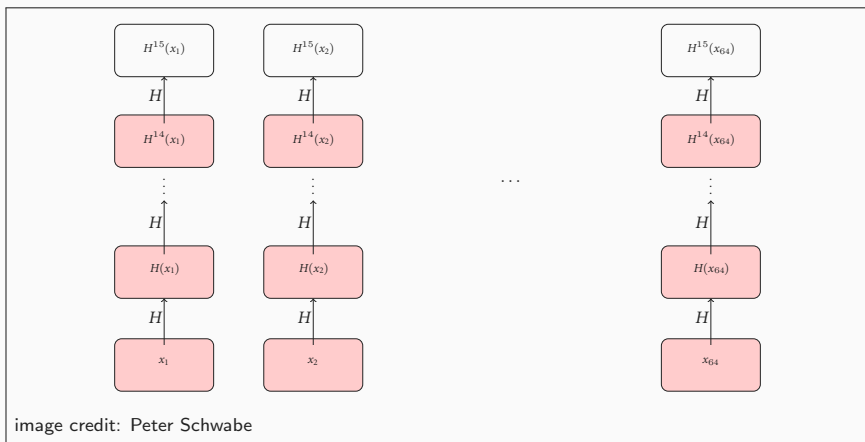
- Generate 256-bit random values $(x_1, x_2, \dots, x_{64})$ (secret key)
- Compute $(y_1, y_2, \dots, y_{64}) = (H^{15}(x_1), \dots, H^{15}(x_{64}))$ (public key)

- **Signing:**

- Chop 256 bit message into 64 chunks of 4 bits $M = (M_1, \dots, M_{64})$
- Compute $\sigma = (\sigma_1, \dots, \sigma_{64}) = (H^{M_1}(x_1), \dots, H^{M_{64}}(x_{64}))$

- **Verification:**

- Check that $y_1 = H^{15-M_1}(\sigma_1), \dots, y_{64} = H^{15-M_{64}}(\sigma_{64})$



- Length of chain - Trade-off between shorter signatures and speed
- Trivial attack - from signature of (\dots, M_i, \dots) obtain signature of $(\dots, M_i + 1, \dots)$
 - Solution: introduce backward chains for checksum values C_j – when M_i is increased, some of the C_j is reduced

Standardization of statefull signatures

- RFC 8391 – XMSS: eXtended Merkle Sigthenature Scheme
 - protection against multi target attacks
 - tight security
 - keys and signatures - a few kilobytes
 - perfect for applications where it is natural to count signatures - updates, code signing
 - stateful means no reuse of OTS key allowed, or going back to previous state!
 - so not good for back-ups, multithreading, VMs
- RFC 8554 – LMS (Leighton-Micali signature)
- NIST endorses XMSS an LMS
- ISO SC27 JTC1 WG2 - study period on stateful hash signatures
- SPHINCS and SPHINCS+ - stateless hash-based signatures
 - idea: make the tree enormous, so that probability of reusing a signatures becomes negligible
 - SPHINCS+ - in the process of standardizing by NIST...draft standard out in Aug '23
 - solid, conservative security, but very big signatures

Today:

- Σ - protocols and Fiat Shamir signatures
- Hash-Based signatures

Next time:

- Authenticated Key-Exchange using Digital Signatures