Digital Signatures Based on Cryptographic Hash Functions

Requirements

- Implementation of the Winternitz one-time signature scheme (WOTS) with exchangeable hash-function
- Implementation of the Merkle signature scheme (MSS) based on WOTS
- Plug-in for the visualization category

Hash Function

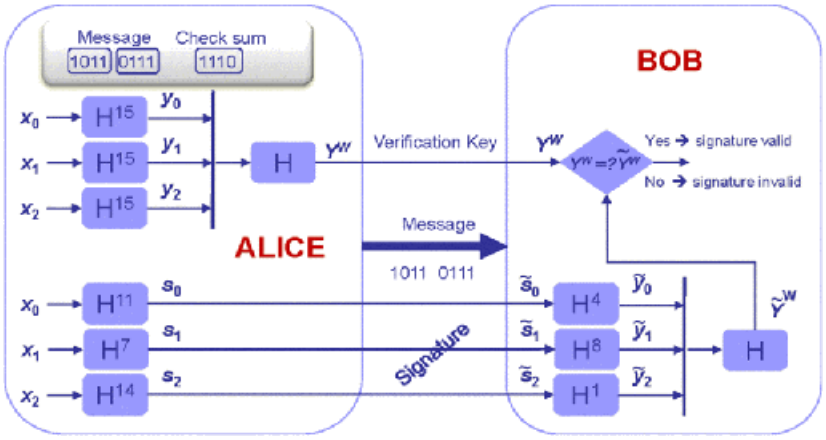
- Mathematical algorithm that maps data of arbitrary size to a bit string of fixed size which is designed to also be a one-way function
- SHA-3 is an example of a hash function

Checksum

- Mathematical value that is assigned to a file and used to "test" the file at a later date to verify that the data contained in the file has not been maliciously changed.
- Created by a complicated series of mathematical operations that translates the data into a hash value.

Winternitz OTS

- Main idea is to iteratively apply a function on a secret input, whereas the number of iterations depends on the number to be signed.
    - The output of the function is if the key for the next iteration, the same input is used for each iteration.
- Generation of the key and digital signature can be found here: https://eprint.iacr.org/2011/191.pdf



Picture of the Winternitz OTS

Require: Winternitz parameter w; Message length and hash value length n.
Ensure: Signature key $X^W \in \{0,1\}^{n \times v}$; Verification key $Y^W \in \{0,1\}^n$.

1: Determine $v = v_1 + v_2$: $v_1 = \lceil \frac{n}{w} \rceil$ (message block number) and $v_2 = \left\lceil \frac{\lfloor log_2 v_1 \rfloor + w + 1}{w} \right\rceil$ (checksum block number);
2: Choose $x_1, \ldots, x_v \in \{0,1\}^n$ uniformly at random;
3: Set $X^W = (x_1, \ldots, x_v)$;
4: Compute $y_i = H^{2^w - 1}(x_i)$ for $i = 1, \ldots, v$;
5: Compute $Y^W = H(y_1 \| \ldots \| y_v)$;
6: **return** (X^W, Y^W).

Algorithm of Key Generation

Merkle Signature Scheme

- Digital Signature Scheme based on hash trees.
- Key generation
  - $N=2^n$ messages
  - Create N Public/Private keys from Winternitz OTS (X_i,Y_i)
  - Create 2^n hashes h_i for from H(Y_i)
  - Make a hash tree with h_i's as the leaves, and each node above is a concat of them
- Signature generation
  - Signer chooses an (X,Y) that hasn't been used to sign yet
  - Include the intermediate nodes of the tree to prove it isn't a newly generated (X,Y)
  - The final signature is the concatenation of the OTS with X_i,Y_i and the hashed nodes+their sibling node
- Signature verification
  - Receiver knows the root of the tree (pub), the message, and the total signature
  - Verify the OTS, compute H(Y_i), and compute H(nodes||sibling).
  - If the result is equal to pub, then the signature is valid
- Full Wikipedia info on it: https://en.wikipedia.org/wiki/Merkle_signature_scheme