Introduction to Cryptology

8.2 - Hash functions: Further Applications

Federico Pintore

Mathematical Institute, University of Oxford (UK)

Michaelmas term 2020

Hash Functions: Additional Applications

Fingerprinting: The digest $H(x)$ of a file x (which could be a virus) acts as a fingerprint/identifier of the file.

Hash Functions: Additional Applications

Fingerprinting: The digest $H(x)$ of a file x (which could be a virus) acts as a fingerprint/identifier of the file.

Deduplication: used to eliminate duplicate copies of data. It is particularly important for cloud storage.

- The hash of the file to store is sent to the service (e.g. DropBox);
- the service checks if a file with this hash already exists;
- if yes, they do not need to store the file again.

Merkle Trees

How to create the fingerprint of a set
of n files $x_1, \cdots, x_n$ (where n is a power of 2)?

Merkle Trees

How to create the fingerprint of a set
of n files $x_1, \cdots, x_n$ (where n is a power of 2)?

Instead of computing i.e. $H(x_1, \cdots, x_n)$, Ralph Merkle proposed the following solution (Merkle Trees):

Merkle Trees

How to create the fingerprint of a set
of n files $x_1, \cdots, x_n$ (where n is a power of 2)?

Instead of computing i.e. $H(x_1, \cdots, x_n)$, Ralph Merkle proposed the following solution (Merkle Trees):

- Compute $h_{1,2} \leftarrow H(x_1, x_2), \cdots, h_{n-1,n} \leftarrow H(x_{n-1}, x_n)$.
- Compute $h_{1,2,3,4} \leftarrow H(h_{1,2}, h_{3,4}), \cdots, h_{n-3,n-2,n-1,n} \leftarrow H(h_{n-3,n-2}, h_{n-1,n})$
- The process is iterated, until the root $h_{1,\ldots,n}$ is computed.

Merkle Trees

How to create the fingerprint of a set
of n files $x_1, \cdots, x_n$ (where n is a power of 2)?

Instead of computing i.e. $H(x_1, \cdots, x_n)$, Ralph Merkle proposed the following solution (Merkle Trees):

- Compute $h_{1,2} \leftarrow H(x_1, x_2), \cdots, h_{n-1,n} \leftarrow H(x_{n-1}, x_n)$.
- Compute $h_{1,2,3,4} \leftarrow H(h_{1,2}, h_{3,4}), \cdots, h_{n-3,n-2,n-1,n} \leftarrow H(h_{n-3,n-2}, h_{n-1,n})$
- The process is iterated, until the root $h_{1,\ldots,n}$ is computed.

Possible alternative to the Merkle-Damgård transform. It is not collision-resistant if n is not fixed.

Password Hashing

A hash of the password is usually stored instead of the password itself.

Password Hashing

A hash of the password is usually stored
instead of the password itself.

- Passwords must not be chosen from a small space.
- To rely on the preimage resistance of a hash function H, passwords must be uniformly sampled.

Password Hashing

A hash of the password is usually stored
instead of the password itself.

- Passwords must not be chosen from a small space.
- To rely on the preimage resistance of a hash function H, passwords must be uniformly sampled.

If passwords are random combinations of 8 alphanumeric characters, the password space has cardinality $N = 62^8 \approx 2^{47.6}$.

- There is an attack (that requires some preprocessing) which only uses time and space $N^{2/3} \approx 2^{32}$.
- Mechanisms to mitigate this threat (long random salt, etc.).

Commitment Schemes

A commitment scheme allows a party to commit to a value v by producing a commitment on it.

Commitment Schemes

A commitment scheme allows a party to commit to a value v by producing a commitment on it.

- The commitment keeps the value v hidden, i.e. it reveals nothing about it (Hiding property).
- The party cannot open the commitment to two different values v_1, v_2 (Biding property).

Commitments Schemes

Definition

A commitment scheme consists of two algorithms, Gen and Commit*, defined as follows:*

- $\mathrm{p} \leftarrow$ *Gen*(n) : *on input a security parameter* n*, it outputs public parameters* p*.*
- $\mathrm{com}_{(m)} \leftarrow \mathrm{Commit}(\mathrm{p}, m \in \{0,1\}^n, r \in \{0,1\}^n)$: *it takes the public parameters, a message* m *and a random value* r*, and outputs* $\mathrm{com}_{(m)}$*.*

Commitments Schemes

Definition

A commitment scheme consists of two algorithms, Gen and Commit*, defined as follows:*

- $\mathrm{p} \leftarrow$ *Gen*(n) : *on input a security parameter* n*, it outputs public parameters* p*.*
- $\mathrm{com}_{(m)} \leftarrow \mathrm{Commit}(\mathrm{p}, m \in \{0,1\}^n, r \in \{0,1\}^n)$: *it takes the public parameters, a message* m *and a random value* r*, and outputs* $\mathrm{com}_{(m)}$*.*

The sender opens their commitment by revealing both m and r.

Commitments Schemes

Definition

A commitment scheme consists of two algorithms, Gen and Commit*, defined as follows:*

- $\mathrm{p} \leftarrow$ *Gen*(n) : *on input a security parameter* n*, it outputs public parameters* p*.*
- $\mathrm{com}_{(m)} \leftarrow \mathrm{Commit}(\mathrm{p}, m \in \{0,1\}^n, r \in \{0,1\}^n)$: *it takes the public parameters, a message* m *and a random value* r*, and outputs* $\mathrm{com}_{(m)}$*.*

The sender opens their commitment by revealing both m and r.

A commitment scheme is secure if it is both binding and hiding.

Commitments Schemes

Let H be a collision resistant hash function. Define a commitment scheme where $H(m||r) \leftarrow \text{Commit}(\text{p}, m, r)$.

Commitments Schemes

Let H be a collision resistant hash function. Define a commitment scheme where $H(m||r) \leftarrow \text{Commit}(\text{p}, m, r)$.

Binding: follows from the collision resistance of H.

Hiding: follows from the uniformity of r in $\{0,1\}^n$ (H modelled as a random oracle).

Commitments Schemes

Let H be a collision resistant hash function. Define a commitment scheme where $H(m||r) \leftarrow \text{Commit}(\text{p}, m, r)$.

Binding: follows from the collision resistance of H.

Hiding: follows from the uniformity of r in $\{0,1\}^n$ (H modelled as a random oracle).

There exist commitment schemes proven secure in the standard model.

Further Reading I


Mihir Bellare and Phillip Rogaway.
Random oracles are practical: A paradigm for designing efficient protocols.
In Proceedings of the 1st ACM conference on Computer and communications security, pages 62–73. ACM, 1993.


Guido Bertoni, Joan Daemen, Michaël Peeters, and Gilles Van Assche.
Keccak sponge function family main document.
Submission to NIST (Round 2), 3:30, 2009.


Jean-Sébastien Coron, Yevgeniy Dodis, Cécile Malinaud, and Prashant Puniya.
Merkle-Damgård revisited: How to construct a hash function.
In Advances in Cryptology–CRYPTO 2005, pages 430–448. Springer, 2005.

Further Reading II


Morris J Dworkin.
SHA-3 standard: Permutation-based hash and extendable-output function.
No. Federal Inf. Process. Stds.(NIST FIPS)-202, 2015.


Pierre Karpman, Thomas Peyrin, and Marc Stevens.
Practical free-start collision attacks on 76-step SHA-1.
In Advances in Cryptology–CRYPTO 2015, pages 623–642. Springer, 2015.


Neal Koblitz and Alfred J Menezes.
The random oracle model: a twenty-year retrospective.
Designs, Codes and Cryptography, pages 1–24, 2015.

Further Reading III


Alfred J Menezes, Paul C Van Oorschot, and Scott A Vanstone.
Handbook of applied cryptography.
CRC press, 1996.


Marc Stevens.
New collision attacks on SHA-1 based on optimal joint local-collision analysis.
In Advances in Cryptology–EUROCRYPT 2013, pages 245–261. Springer, 2013.


Marc Stevens, Elie Bursztein, Pierre Karpman, Ange Albertini, and Yarik Markov.
The first collision for full SHA-1.
In Annual International Cryptology Conference–CRYPTO 2017, pages 570–596. Springer, CHam, 2005.