NYU

DS-GA 3001.009 Responsible Data Science Lab 5

Center for Data Science
Udita Gupta | Tandon School of Engineering

Hashing and Salting

Encryption/ Hashing/ Salting?

- **Encryption**
  - Encryption is the practice of scrambling information in a way that only someone with a corresponding key can unscramble and read it.
  - It's a **two-way function**.
- **Hashing**
  - Hashing is the practice of using an algorithm to map data of any size to a fixed length.
  - This is called a hash value (or sometimes hash code or hash sums).
  - It's a **one-way function**.
- **Salting**
  - Salting is an additional step during hashing, typically seen in association to hashed passwords, that adds an additional value to the end of the password that changes the hash value produced.

Properties

- Computationally Efficient
- Deterministic
- Pre-Image Resistant
- **Output of a hash function** always has the **same size**.
  - Ex: SHA-256 has output of 256 bits (64 hexadecimal) (4 bits : 1 hex)
- Collision Resistant

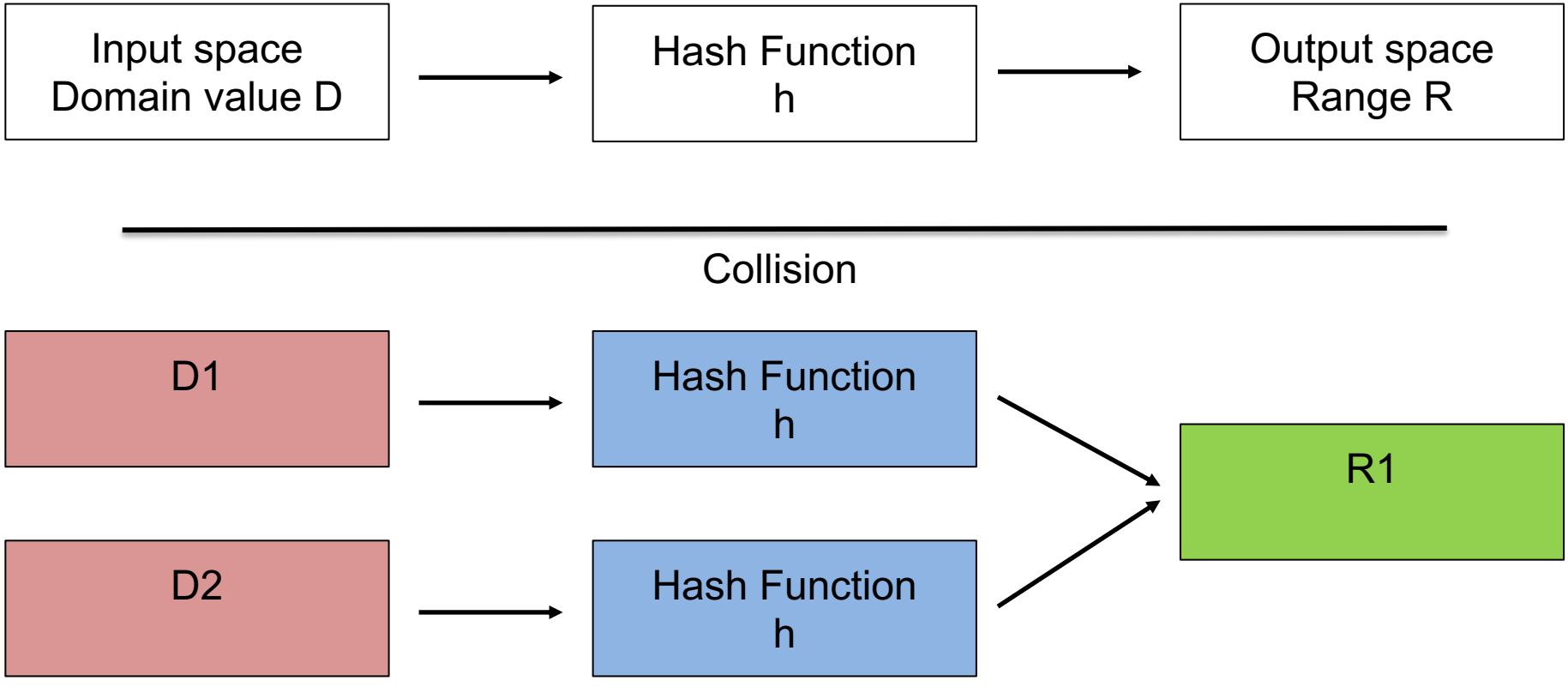
Hashing

- One-way functions
- We have a hashing function h. It takes a domain of values (input space, call it D) and maps them to a smaller range of values (output space, call it R).
- Given d1, d2 in D and both map to r1 and r2 (both in R) respectively.
  - h(d1) = r1, h(d2) = r2
  - We would define collision as h(d1) = r1, h(d2) = r2 where r1== r2.
  - The probability of a collision, depends on two things:
    - on how good the hashing function is
    - on the relative sizes (number of distinct values) in the domain and in the range.

Hashing

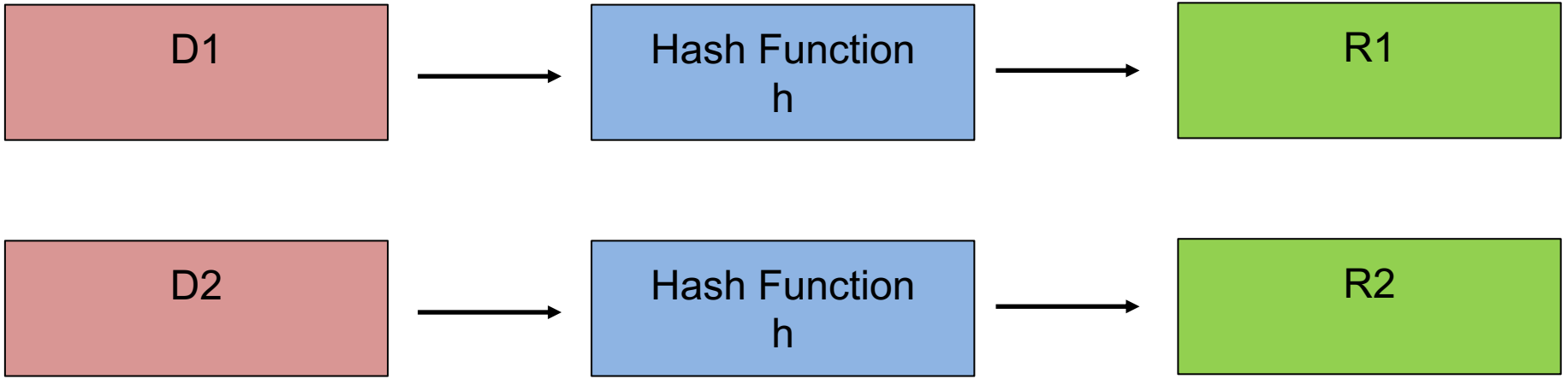
- Note:
  - If two values in the domain (d1, d2) map to different values in the range (h(d1) = r1, h(d2) = r2, and r1 != r2), then we know that d1 != d2. This is because the hashing function h is deterministic.
  - However, the **converse is not true**: If two values d1 != d2 in the domain map to the same value in the range (h(d2) = r1, h(d2) = r2, and r1=r2), it's still possible that the domain values are the same, but it's not guaranteed.
  - Consequently, "breaking" a hash function does not actually guarantee that you reverse-engineered the true value in the input (in the domain).
  - It just allows you to guess the value with a high probability, assuming that you have a good hashing function and that the probability of a collision is rare.
  - What the probability actually is, depends on the hashing function, and on what you know about the data distribution in the input.

Hashing


Input space
Domain value D
Hash Function
h
Output space
Range R
Collision
D1
Hash Function
h
D2
Hash Function
h
R1

Hashing

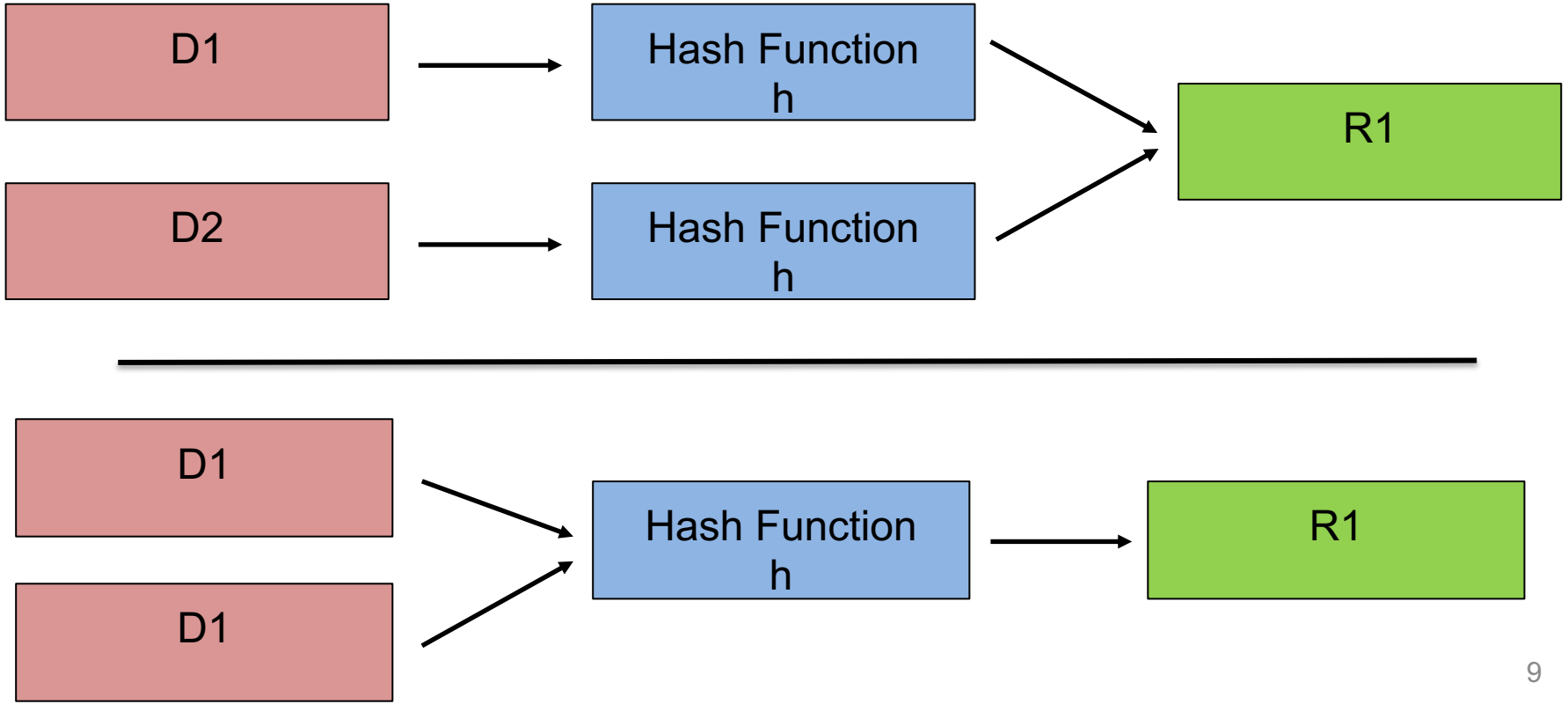
If two values in the domain (d1, d2) map to different values in the range (h(d1) = r1, h(d2) = r2, and r1 != r2), then we know that d1 != d2.



Hashing - Collision

However, converse is not true.

Given, h(d1) = r1, h(d2) = r2, and r1 == r2), we can not guarantee that D1 == D2



SHA

- **SHA : Secure Hashing Algorithm**
- **SHA-1 and SHA-2** are two different versions of that algorithm.
- They differ:
  - **Construction** (how the resulting hash is created from the original data)
  - **Bit-length** of the signature.
  - SHA-2 is like the successor to SHA-1, as it is an overall improvement.
  - SHA-1 is a 160-bit hash.
  - SHA-2 is actually a “family” of hashes and comes in a variety of lengths, the most popular being **256-bit**.
- The SSL industry has picked SHA as its **hashing algorithm for digital signatures**.

SHA-2

- **SHA-2:“SHA-224,” “SHA-384,” or “SHA-512,”** those are referring to the alternate bit-lengths of SHA-2.

- Most commonly accepted is SHA-256.

- **SSL:**
  - From 2011 to 2015, **SHA-1 was the primary algorithm.**
  - Google has even gone so far as to create a **SHA-1 collision** (when two pieces of disparate data create the same hash value).
  - From 2016 onward, **SHA-2 is the new standard.**

How many hashes?

- A **larger bit hash** can provide more security because there are **more possible combinations.**
  - *Remember*: One of the important functions of a cryptographic hashing algorithm is that is produces unique hashes.
  - If two different values or files can produce the same hash, you create what we call a **collision**.

- If a hashing algorithm is supposed to produce unique hashes for every possible input, just how many possible hashes are there?
  - A bit has two possible values: 0 and 1
  - The possible number of unique hashes can be expressed as the number of possible values raised to the number of bits.
  - For SHA-256 there are $\mathbf{2^{256}}$ possible combinations

- *Note*: A large bit-length does not automatically mean a hashing algorithm produces more secure hashes.
  - The **construction** of the algorithm is also incredibly important – that's why the SSL industry uses hashing algorithms specifically designed for cryptographic security.

Salting

- Salting is a concept that typically pertains to **password hashing**.
- Essentially, it's a unique value that can be added to the end of the password to create a different hash value.
- This **adds a layer of security** to the hashing process, specifically against brute force attacks.
- When salting, the **additional value** is referred to as a **"salt".**

Salting

- You can add “salt” to the **start or end** of your passwords.
- Example:
  - The password I want to salt looks like this: 7X57CKG72JVNSSS9
  - Your salt is just the word **SALT**
  - Before hashing, you add SALT to the end of the data. So, it would look like this: 7X57CKG72JVNSSS9SALT
- The hashed value is different than it would be for just the plain unsalted password.
  - Remember, even the slightest variation to the data being hashed will result in a different unique hash value.

Questions?

Thank you