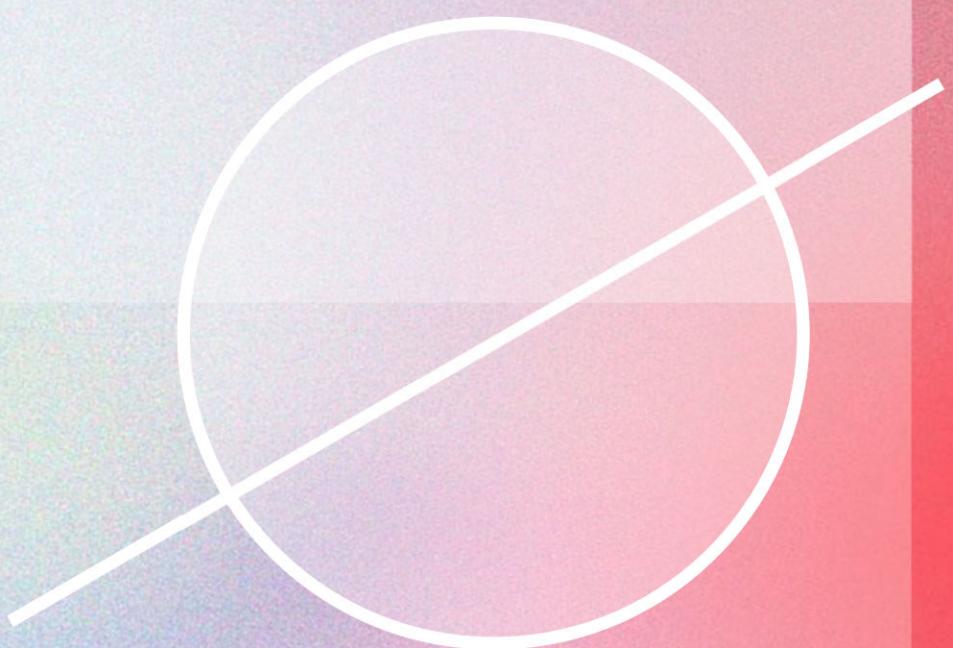


HASH FUNCTIONS



Content Curated by Pollux M. Rey

FOR THIS UNIT...

01

CRYPTOGRAPHIC
HASH FUNCTIONS

02

PROPERTIES

03

APPLICATIONS

04

SECURITY

FOR THIS UNIT...

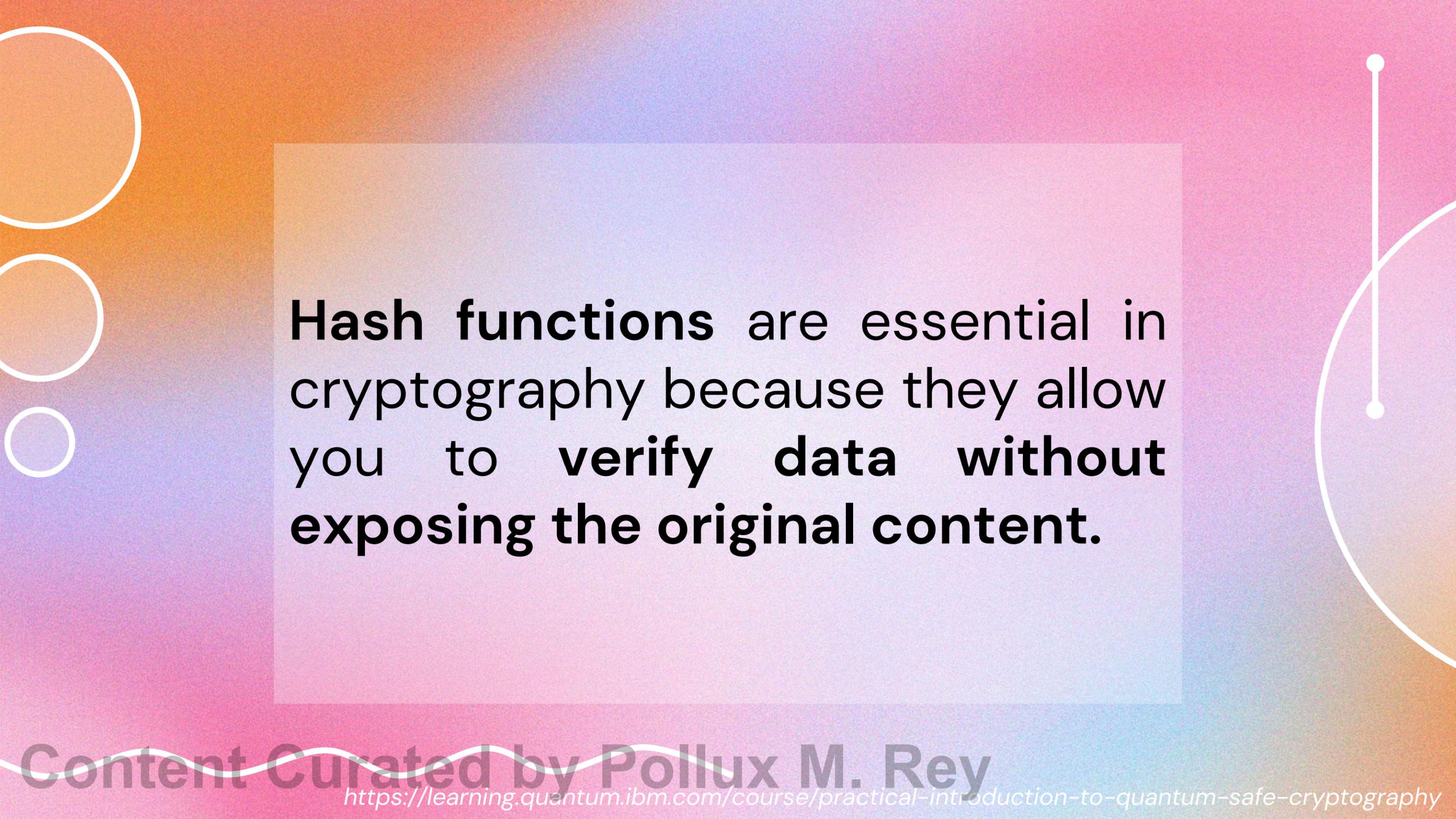
05

**COMMONLY USED
CHFs**

Content Curated by Pollux M. Rey

01

WHAT IS CRYPTOGRAPHIC HASH FUNCTIONS?



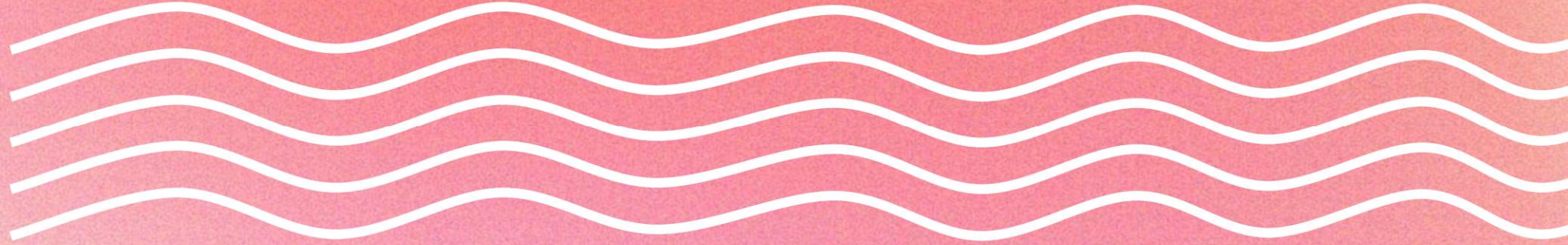
Hash functions are essential in cryptography because they allow you to **verify data without exposing the original content.**



They play a crucial role in ensuring both **data authenticity** and **integrity**.



02



BASIC RATIONALE AND DESIGN OF HASH FUNCTIONS



There are many situations where **authentication** and **integrity verification** need to be performed cheaply and without revealing private information to the party performing the validation.

There are many situations where **authentication** and **integrity verification** need to be performed cheaply and without revealing private information to the party performing the validation.

EXAMPLES



***Verifying that downloaded software
hasn't been modified.***

Windows 10 / 11



[Download qBittorrent v5.0.4](#) (multiple installer choice)
► (Additional download options)

[Download qBittorrent v5.1.0rc1](#) (multiple installer choice)
► (Additional download options)

▼ Checksums and library versions

Version	SHA2-256
5.0.4	61e516ba3be4ff0f3fd226d6271c5f7a505c2a5b080ab28c4964f741b569b1e4
5.0.4 (qt6 lt20)	e715ce2d484bd27073e0be6837fd72c97ab604b5848f529c9901eb02cc4a7245
5.1.0rc1	9388eb9e683ab209da9df8709b5d1cedf1154437e0787f74e0516e275d10185b
5.1.0rc1 (lt20 qt6)	08d494e793246a989b7110ea204f0da7ecfbec943624db2b2e1b782aae08a09

Library Version
libtorrent-rasterbar 1.2.20+git2e537ee7af / 2.0.11+gitc31d90b59f
Qt 6.7.3 (v5.0.4) / 6.8.2 (v5.1.0rc1)
Boost 1.86.0

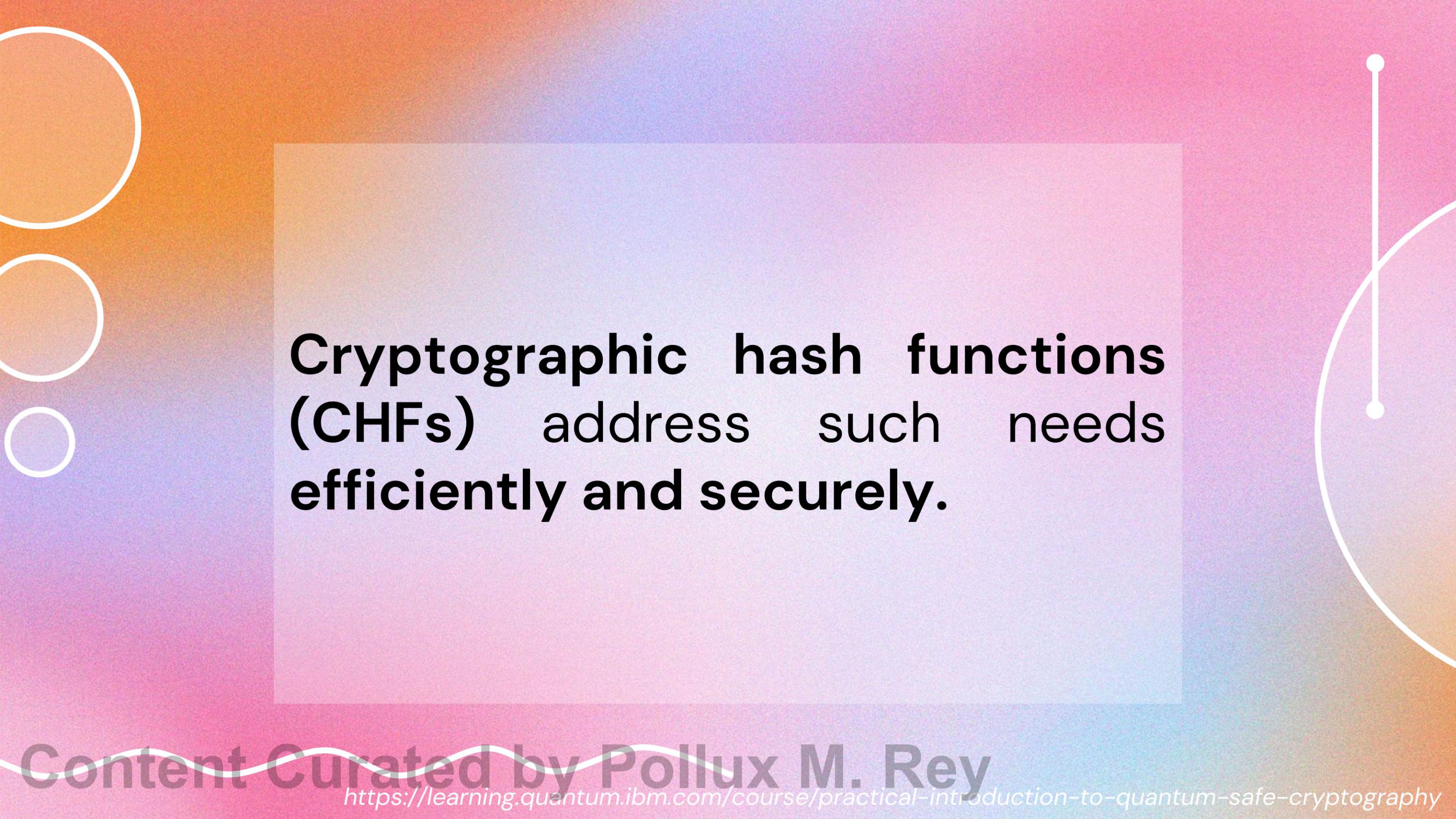
[Download nightly builds](#)

EXAMPLES



Authenticating web users without storing or transmitting actual passwords.

```
{  
  "_id": 31498,  
  "email": "rey.pollux@marsu.edu.ph",  
  "password": "$2b$12$rkEH4w05PyDqjJPJDUA07eLQZ53u2LiGzfgf5LCci.XVe5tYrFdT0",  
  "temporaryPassword": [REDACTED]  
  "firstName": "Pollux",  
  "middleName": "Murillo",  
  "lastName": "Rey"  
}
```

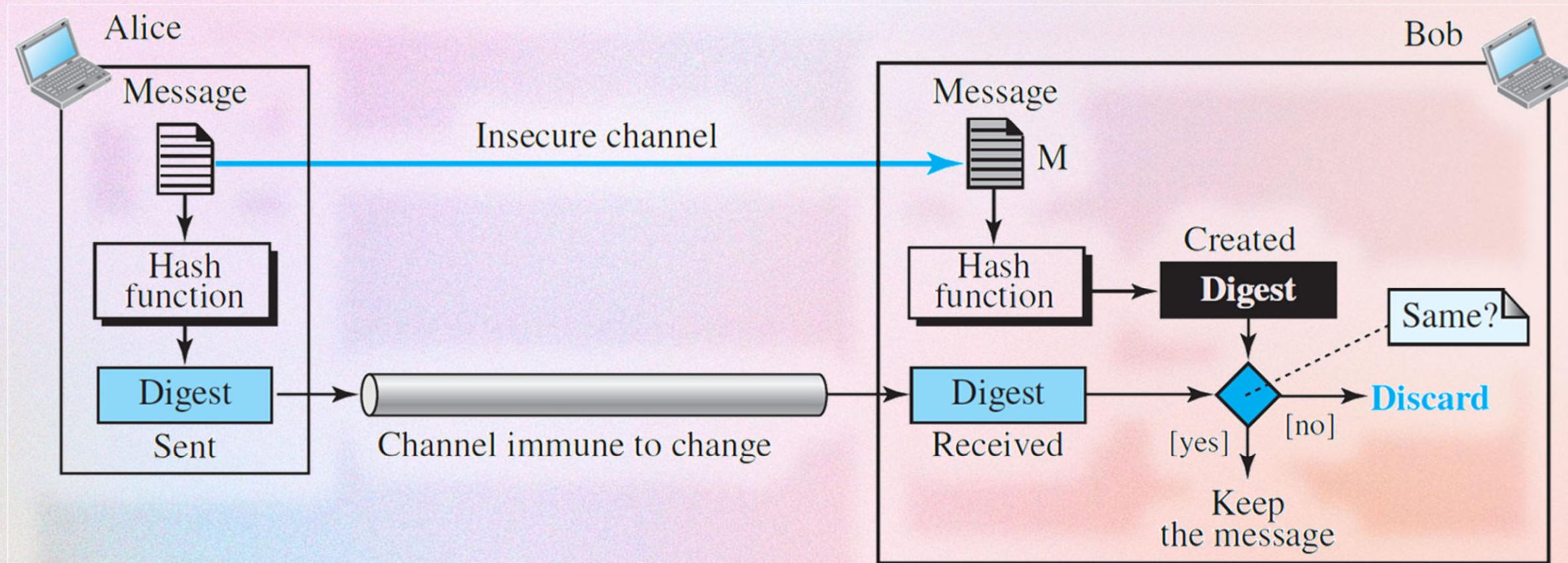


**Cryptographic hash functions
(CHFs) address such needs
efficiently and securely.**

CRYPTOGRAPHIC HASH FUNCTIONS



A cryptographic hash function **converts any message into a fixed-length n-bit string called a digest.**



PROPERTIES OF CHFs



A useful CHF should satisfy several key properties:

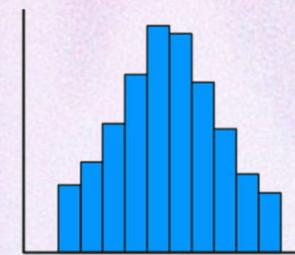
1. Uniformity
2. Determinism
3. Irreversibility
4. Approximate Injectivity

UNIFORMITY

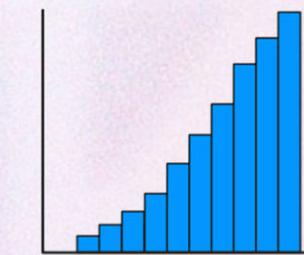


The output of a CHF should be distributed uniformly and appear random, ensuring that no clues about the input are revealed.

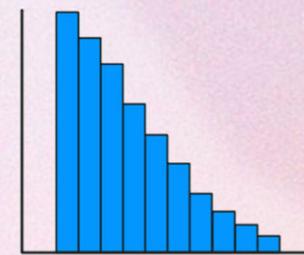
Histogram Distributions



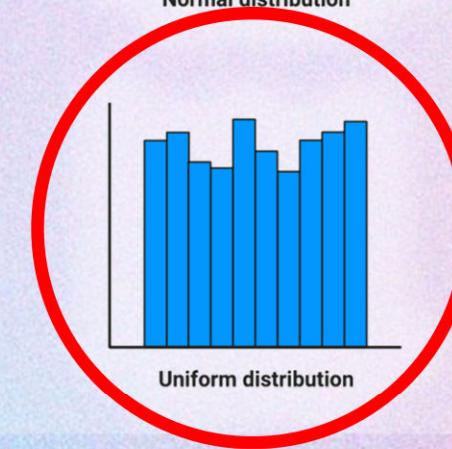
Normal distribution



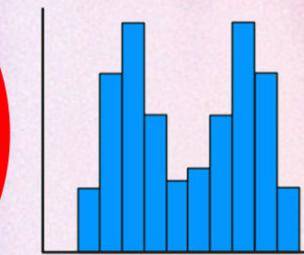
Left-skewed distribution



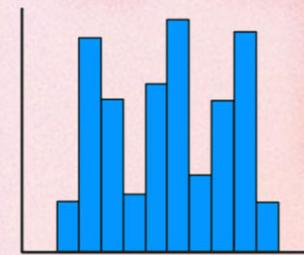
Right-skewed distribution



Uniform distribution



Bimodal distribution



Multimodal distribution

UNIFORMITY



The output of a CHF should be distributed uniformly and appear random, ensuring that no clues about the input are revealed.

Example:

If the MD5 hash **8b1a9953c4611296a827abf8c47804d7** (which corresponds to "Hello") can be generated from 1,000,000 different messages, then other hashes should also be generated from about 1,000,000 different messages.

DETERMINISM



The same input must always produce the same digest.

Example:

SHA256 File Checksum

This SHA256 online tool helps you calculate the hash of a file from local or URL using SHA256 without uploading the file. It also supports HMAC.

Settings

Hash

Auto Update

Remember Input

Input Type

File

Output Encoding

Hex (Lower Case)

Enable HMAC

Input

Drag and drop the file here or click to select a file. It will process locally and won't be uploaded.



Output

Output here...

Content Curated by Pollux M. Rey

<https://learning.quantum.ibm.com/course/practical-introduction-to-quantum-safe-cryptography>

IRREVERSIBILITY



It should be nearly impossible to reverse the process and retrieve the original input from the output.

Counterexample:



Malicious Life Podcast: What The LinkedIn Hack Taught Us About Storing Passwords

WRITTEN BY Malicious Life Podcast

In June 2012, an anonymous hacker posted a list of 6.5 Million encrypted passwords belonging to LinkedIn users on a Russian hacker forum. It was soon discovered that these passwords were hashed using an outdated and vulnerable hashing algorithm and were also unsalted.

The lawsuits followed suit shortly. What is 'hashing' and 'salting' and can we trust big organizations to keep our secrets safe? Check it out...

Search



SUBSCRIBE
Never miss a blog.

IRREVERSIBILITY



The screenshot shows a web-based tool for SHA1 encryption and decryption. On the left is a dark sidebar menu with white icons and text:

- Home
- Encrypt / Decrypt (highlighted with an orange border)
- Obfuscation
- Ciphers
- Conversion
- Network
- API
- Contact
- FR / EN

The main area has a light gray background. At the top center is the title "Sha1 Encrypt & Decrypt" next to a padlock icon. Below the title is a text input field with placeholder text: "Enter one or several SHA1 hash(es) (max. 100)". To the right of the input field is a larger text area labeled "Output...". At the bottom of the main area are two dark blue buttons: "Encrypt" on the left and "Decrypt" on the right.

Content Curated by Pollux M. Rey
<https://learning.quantum.ibm.com/course/practical-introduction-to-quantum-safe-cryptography>

<https://md5decrypt.net/en/Sha1/>

APPROXIMATE INJECTIVITY



While CHFs are many-to-one functions, they should appear to behave like one-to-one functions.

Small input changes should produce vastly different digests.

APPROXIMATE INJECTIVITY



Example:

Input:
I LOVE CICS

SHA-256 Digest:
**7255aca5ea60dece819eb67de70539bb
b0c51d43b825841a51da315bddc0bd8a**

APPROXIMATE INJECTIVITY



Example:

Input:
I LOVE SICS

SHA-256 Digest:
**ed12a878e5c9f97a3da0c6bd9c07c252
59c0cafbcce897ac653747666ebe4057**

APPROXIMATE INJECTIVITY



Example:

7255aca5ea60dece819eb67de70539bb

B0c51d43b825841a51da315bddc0bd8a

Ed12a878e5c9f97a3da0c6bd9c07c252

59c0cafbcce897ac653747666ebe4057

Character difference: **59**

Given this, it's possible to validate a piece of data against the original instance by comparing a digest of the data to a digest of the original.

CRYPTOGRAPHIC HASH FUNCTIONS



1. If the two digests match, we can be confident with high probability that the data is identical to the original.
2. If the digests differ, we can be sure that the data was tampered with or is otherwise inauthentic.

Since the CHF digests themselves do not reveal the actual contents of the data or the original, they enable validation while preserving privacy.

03

APPLICATIONS OF CRYPTOGRAPHIC HASHING

Content Curated by Pollux M. Rey

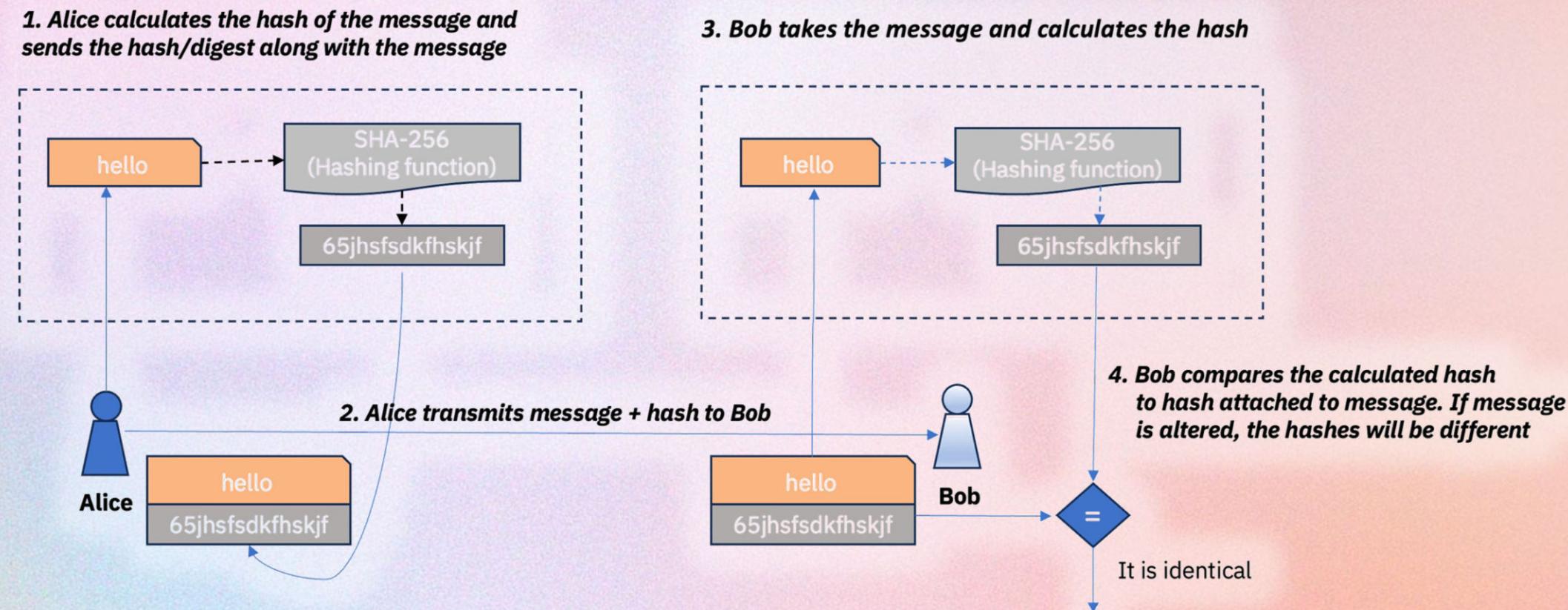


The unique properties of CHFs make them suitable for a wide array of applications.

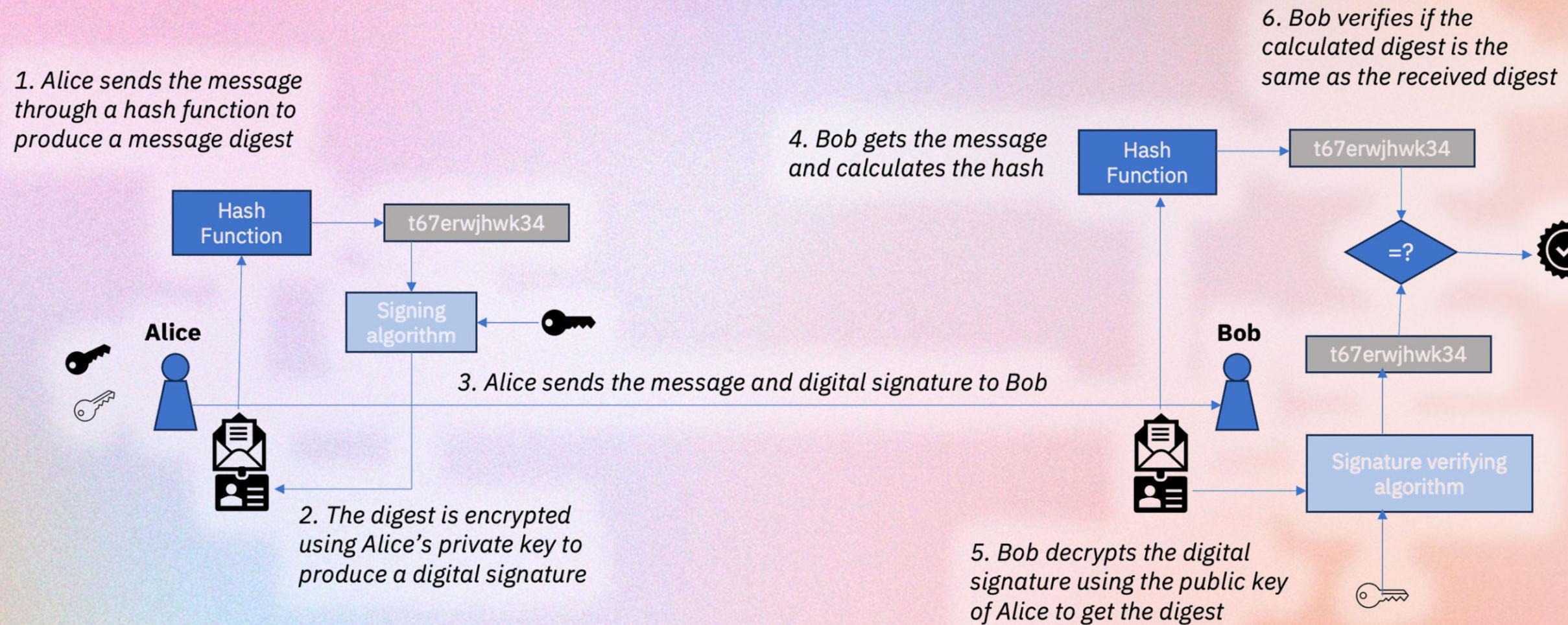
DATA INTEGRITY CHECKS



Any modifications to the data, intentional or not, will result in a different checksum, alerting systems or users to the change.



DIGITAL SIGNATURES



BLOCKCHAIN AND CRYPTOCURRENCIES



Cryptocurrencies like Bitcoin rely heavily on CHFs, particularly in creating transaction integrity and enabling consensus mechanisms like proof of work.

BLOCKCHAIN AND CRYPTOCURRENCIES



Content Curated by Pollux M. Rey

https://www.youtube.com/watch?v=_16OoMzbIY8

04

SECURITY OF CRYPTOGRAPHIC HASHING

Content Curated by Pollux M. Rey

The **security** of a CHF is typically assessed based on resistance to two types of attacks: and pre-image and collision.

PRE-IMAGE RESISTANCE

Pre-image resistance means that given a digest, it should be **infeasible to find the input.**

PRE-IMAGE RESISTANCE

A good CHF is designed in such a way that a party wishing to conduct a **pre-image attack** has **no better option than a brute force approach**, which has time complexity 2^n .

COLLISION RESISTANCE

Collision resistance means that it is difficult to find two different inputs that hash to the same digest.

COLLISION RESISTANCE

Collision resistance is crucial for applications like **digital signatures** and **certificates**, where it could be disastrous if a malicious party were able to create a forgery that hashes to the same value.

COLLISION RESISTANCE

Collision resistance is a harder requirement than pre-image resistance and necessitates output lengths twice as long as that needed for pre-image resistance.

COLLISION RESISTANCE

This is because a brute force attack known as the **birthday attack**, which can be used to identify hash collisions, has time complexity $2^{n/2}$.

05

COMMONLY USED CRYPTOGRAPHIC HASH FUNCTIONS

COMMONLY USED CHFs



Hash Function	Output Length (bits)	Common Applications
MD5	128	File integrity checking, older systems, non-crypto uses
SHA-1	160	Legacy systems, Git for version control
SHA-256	256	Cryptocurrency (Bitcoin), digital signatures, certificates
SHA-3	Variable (up to 512)	Various cryptographic applications, successor to SHA-2
Blake2	Variable (up to 512)	Cryptography, replacing MD5/SHA-1 in some systems
Blake3	Variable (up to 256)	Cryptography, file hashing, data integrity



A graphic design featuring a pink-to-white gradient background with white wavy lines at the top and bottom. In the center-left, a yellow-to-white gradient rectangular area contains the text "THANK YOU!". To the right, there are three overlapping white-outlined circles of increasing size. A thin white horizontal line with small circular caps extends from the left edge of the yellow area to the right edge of the circles.

**THANK
YOU!**

Content Curated by Pollux M. Rey