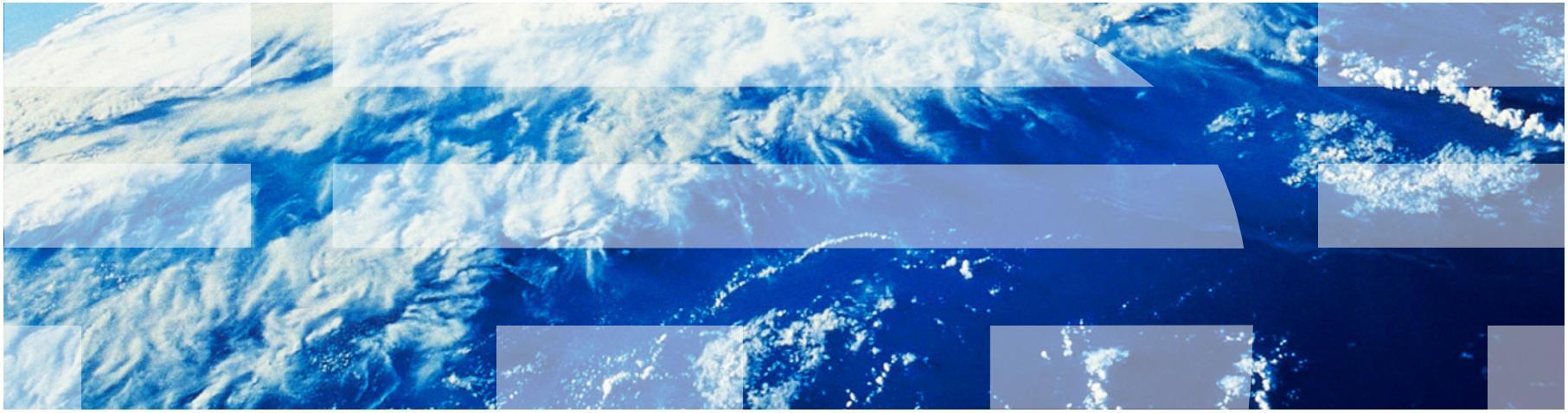
Logistics

- We will release HW3 tonight
  - It will be due on April 14th
  - This is the larger homework with 10% grade value
- Will try to finish midterm grading next week.

Today

- Lecture 7 leftovers
  - Regex and Spark
- Security and privacy
  - How to define security and privacy
    - Confidentiality, integrity, availability, authenticity
- Basic introduction to cryptography
  - Symmetric encryption
  - Asymmetric encryption
  - Digital signatures
- Compliance
  - Types of compliance
  - Difference between security and compliance
- Public Key Infrastructure
- Real-world security tips and tricks

Detour: Regex



Regular expressions (Regex)

- A scripting language for matching and/or manipulating strings of text
  - Manipulating text with built-in Python string functions can be cumbersome
  - Python and PySpark support regex
- Examples of operations
  - Find all substrings with “helloworld” and replace with: “Hello World”
  - Find all substrings with “fox”, but exclude “foxtrot”, and replace with the word ”wolf”
  - Remove all special characters (e.g., !%&), punctuation (e.g., .,;:)
- Useful in many programming contexts
  - Spark
  - Python
  - SQL
  - Etc.
- Hint: can be useful in homework assignment (coupled with a UDF perhaps?)
- Use free online regex tool for debugging: https://regex101.com/

Regex matching examples

Example text: fox foxtrot ox box

fox: fox foxtrot ox box

ox: fox foxtrot ox box

fox\w: fox foxtrot ox box (\w matches any word character)

fox\w*: fox foxtrot ox box (* matches zero or more of the preceding character)

fox\w+: fox foxtrot ox box (+ matches one or more of the preceding character)

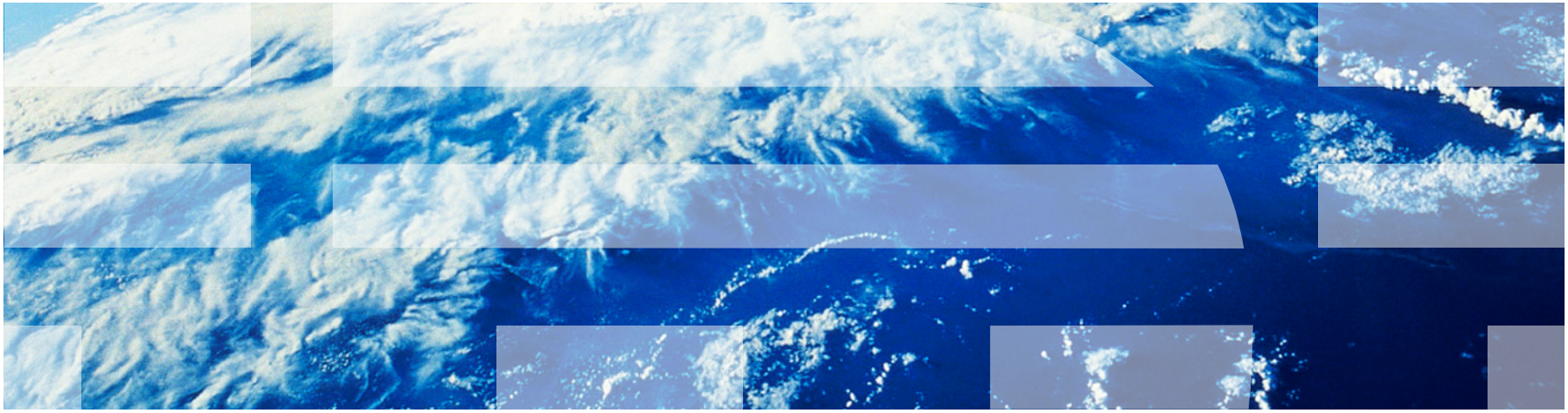
[^f]ox: fox foxtrot ox box (without the letter f in the beginning)

[^f\W]ox: fox foxtrot ox box (without the letter f or a non-word character)

[bf]ox: fox foxtrot ox box (match on b or f followed by ox)

Borrowed from Tathagata Das and Indranil Gupta

Stream Processing



Motivation: ”Hybrid” Between OLAP and OLTP

- Large amounts of data => Need for real-time views of data
  - Social network trends, e.g., Twitter real-time search
  - Website statistics, e.g., Google Analytics
  - Intrusion detection systems, e.g., in most datacenters
- Process large amounts of data
  - With latencies of few seconds
  - With high throughput
- Not exactly OLTP, because updates don’t happen in place (e.g., on rows in table)
- Not exactly OLAP, because it is real time, updates are granular
- Streaming is a relatively new class of data system

Would MapReduce or normal Spark work?

- **Batch Processing** => need to wait for entire computation on large dataset to complete
- Not intended for long-running and real-time stream-processing

Examples of stream processing jobs

A) Uber
   Calculating surge prices

B) LinkedIn
   Aggregating updates into one email

C) Netflix
   Understanding user behavior to improve personalization

D) TripAdvisor
   Calculating earnings per day & fraud detection

Discretized Stream Processing

- Run a streaming computation as a **series of very small, deterministic batch jobs**

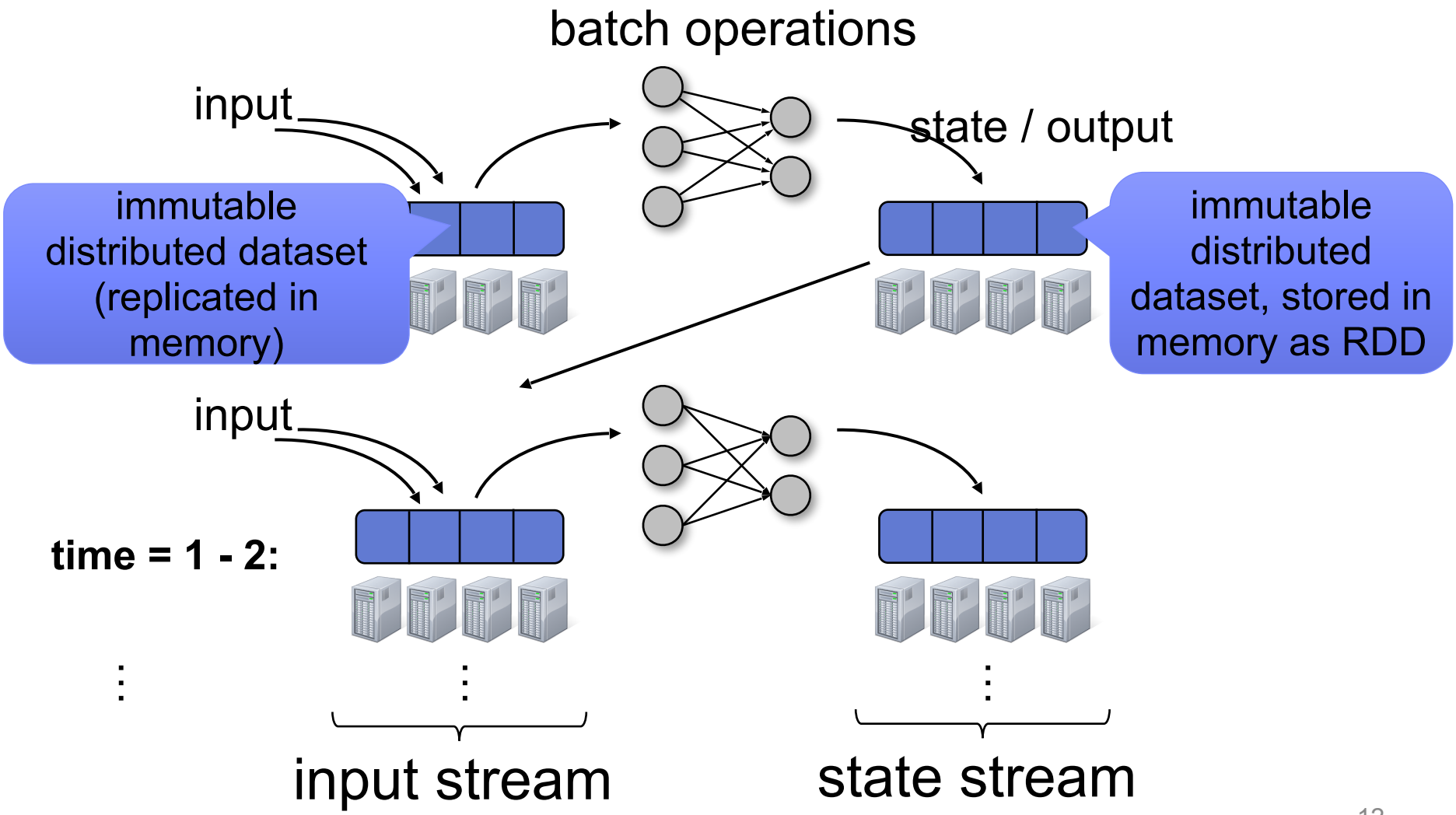
- Batch processing models, like MapReduce, recover from faults and stragglers efficiently
  - Divide job into deterministic tasks
  - Rerun failed/slow tasks in parallel on other nodes

- Same recovery techniques at lower time scales
  - Transformations are not lost (or performed twice) if a worker dies

Spark Streaming

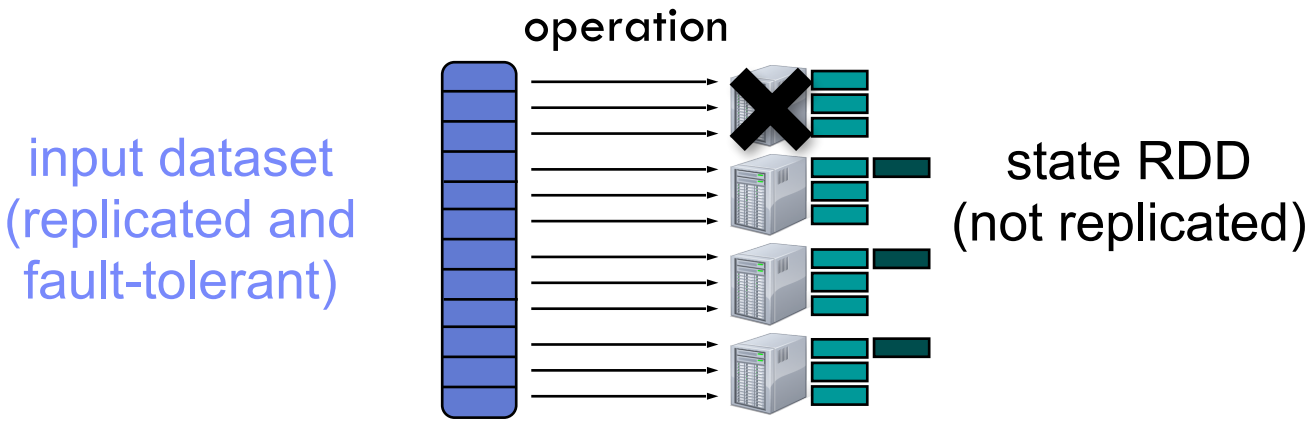
- State between batches kept in memory in **fault-tolerant dataset**
  - Specifically as an RDD/Dataframe/Dataset
- Batch sizes can be reduced to as low as 1/2 second to achieve ~ 1 second latency
- Combines streaming and batch workloads
- Many other alternatives:
  - Apache Storm
  - Apache Flink
  - Amazon Kinesis
  - Google Dataflow
  - …

Discretized Stream Processing



Fault Recovery

- State stored as RDD
  - Deterministically re-computable parallel collection
  - Remembers lineage of operations used to create them
- Fault / straggler recovery is done **in parallel** on other nodes



Fast recovery from faults without full data replication

Programming Model

- A Discretized Stream or **DStream** is a series of RDDs representing a stream of data
  - API *very similar* to RDDs

- DStreams can be created…
  - Either from live streaming data
  - Or by transforming other DStreams

DStream Data Sources

- Many data sources can be inupts
  - HDFS
  - Kafka
  - Flume
  - Twitter
  - …

Transformations

Build new streams from existing streams

- Filters/aggregate operations
  - map, flatMap, filter, count, reduce,
  - groupByKey, reduceByKey, sortByKey, join
  - etc.
- New window and stateful operations
  - window, countByWindow, reduceByWindow
  - countByValueAndWindow, reduceByKeyAndWindow
  - updateStateByKey
  - etc.

Output Operations

Send data to outside world

- saveAsHadoopFiles
- print – prints on the driver's screen
- foreach - arbitrary operation on every RDD

Example

Process a stream of Tweets to find the 20 most popular hashtags in the last 10 mins, updated every 1 second

1. Get the stream of Tweets and isolate the hashtags
2. Count the hashtags over 10 minute window, updated every 1 second

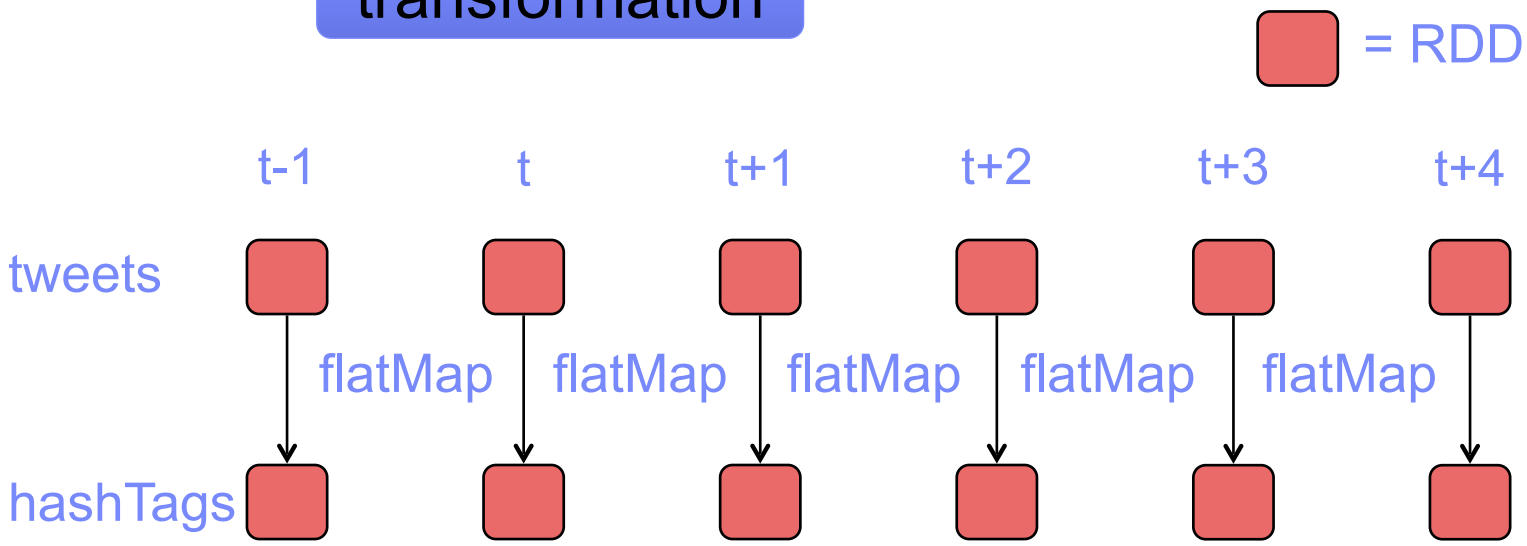
1. Get the stream of Hashtags

```
tweets = ssc.twitterStream(username, password)
```

DStream

```
hashtags = tweets.flatMap (lambda tweet: tweet.getTags())
```

transformation

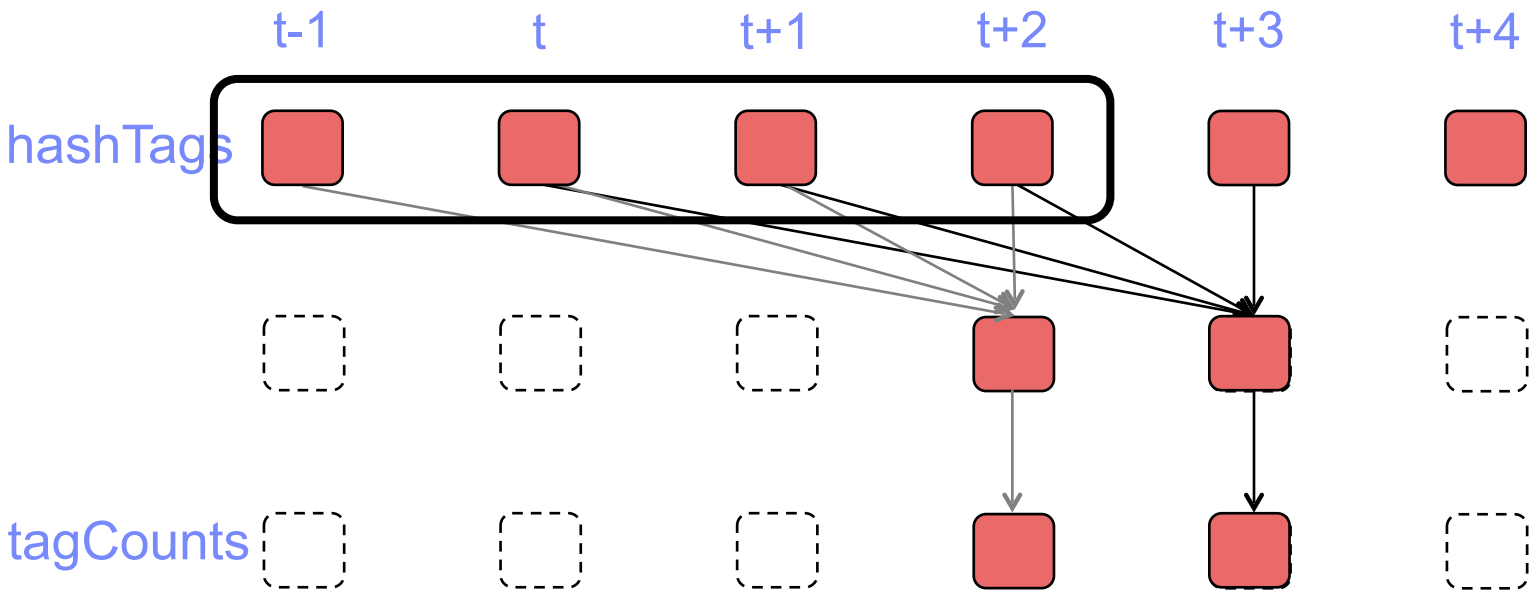


2. Count the hashtags over 10 min

```
tweets = ssc.twitterStream(userna
hashtags = tweets.flatMap (lambda t
tagCounts = hashTags.window(Minutes(10), Seconds(1)).countByValue()
```

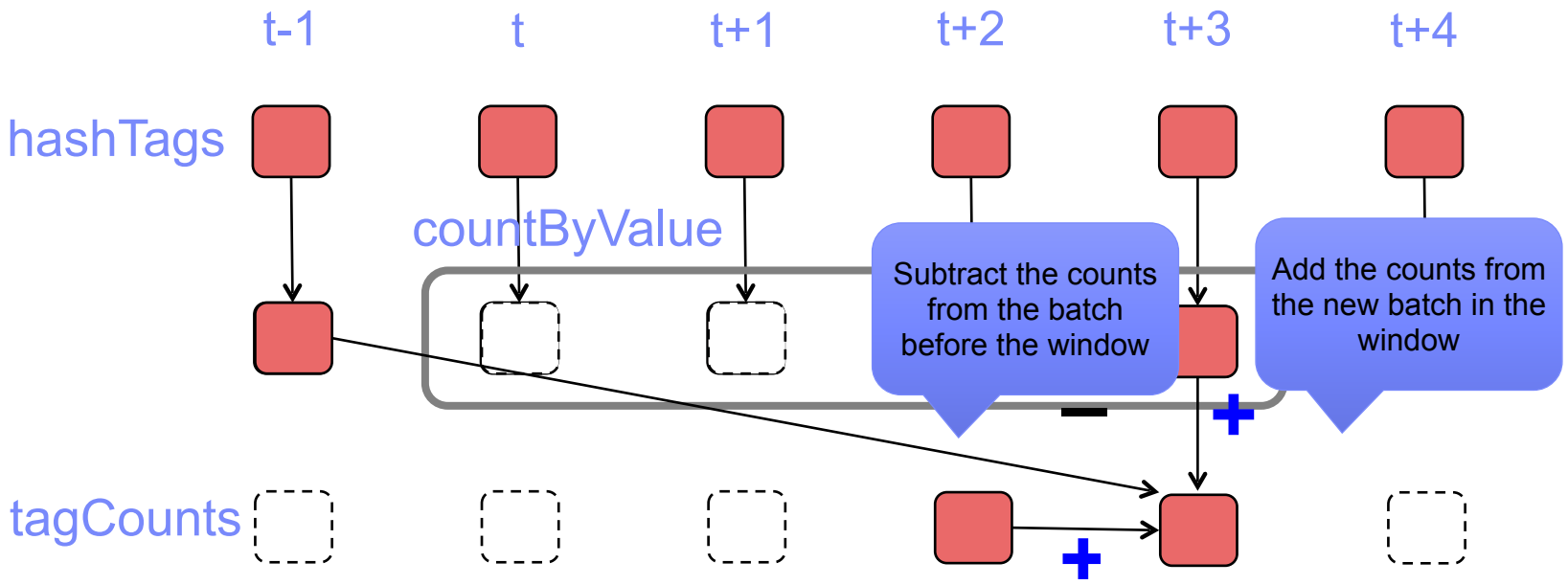
Window length

Sliding interval

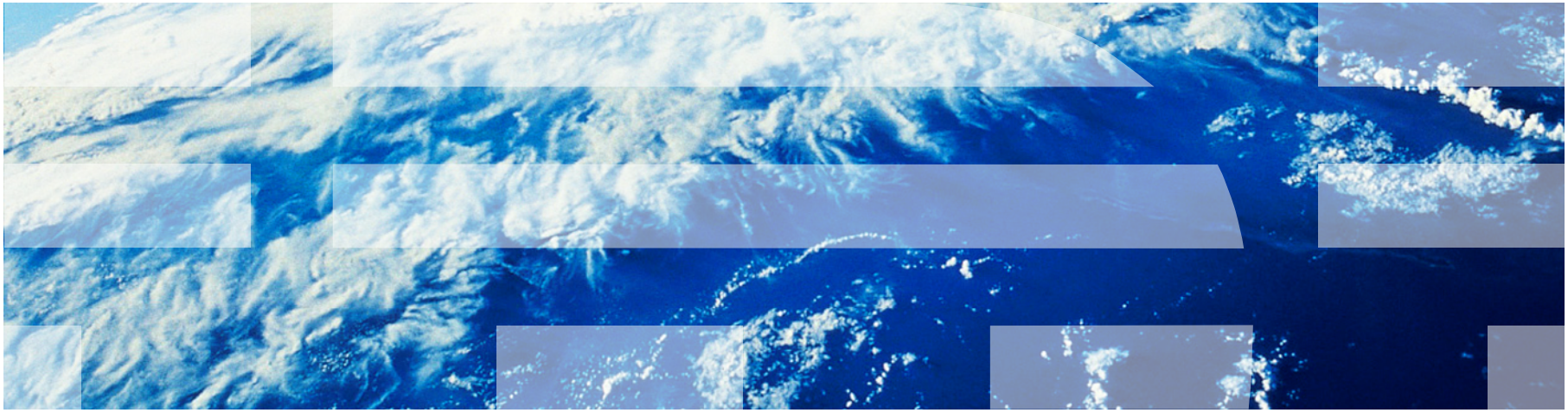


2. Count the hashtags over 10 min

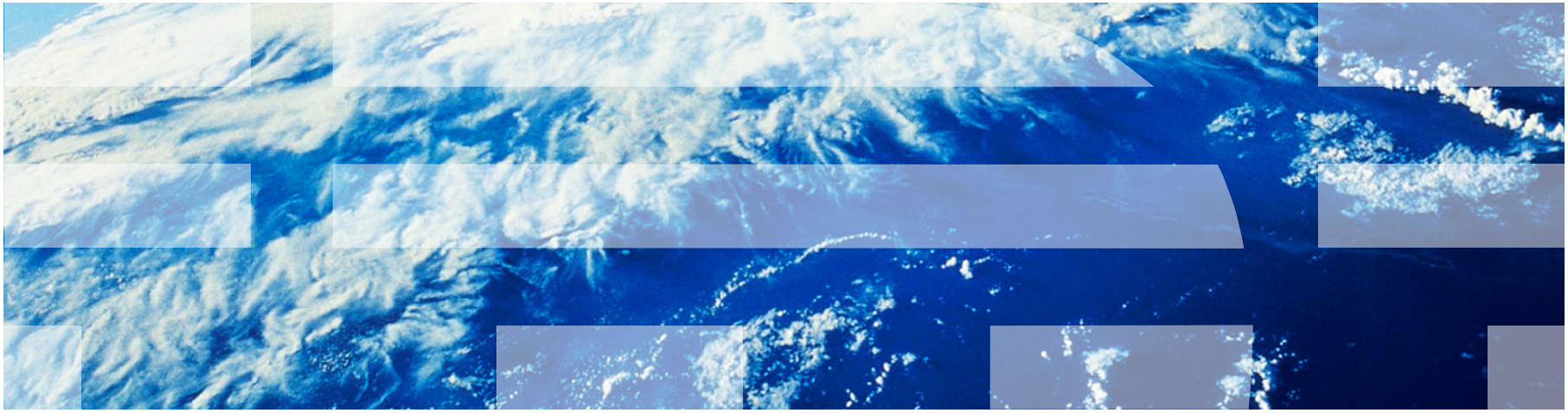
```
tweets = ssc.twitterStream(username, password)
hashtags = tweets.flatMap (lambda tweet: tweet.getTags())
tagCounts = hashtags.countByValueAndWindow(Minutes(10), Seconds(1))
```



Data Security
Compliance



Security Goals



Four security goals

- Confidentiality
  - My secret data was not leaked/stolen
    - E.g., the government, my spouse, a hacker did not gain access to my private information
- Integrity
  - Data / system was not tampered with
    - E.g., nobody added another ‘0’ in their bank account entry
    - E.g., nobody changed the content of my folder without me noticing
- Availability
  - Data or system will be available when needed
    - E.g., my website/service is protected from a malicious hacker trying to take them with a denial of service attack
- Authenticity
  - Belief in the source of the data
    - E.g., this new COVID study came from a reputable journal
    - E.g., the person communicating with me is who they say they are

Cryptography can help address most of these goals

- **Confidentiality**
  - My secret data was not leaked/stolen
    - E.g., the government, my spouse, a hacker did not gain access to my private information
- **Integrity**
  - Data / system was not tampered with
    - E.g., nobody added another ‘0’ in their bank account entry
    - E.g., nobody changed the content of my folder without me noticing
- Availability
  - Data or system will be available when needed
    - E.g., my website/service is protected from a malicious hacker trying to take them with a denial of service attack
- **Authenticity**
  - Belief in the source of the data
    - E.g., this new COVID study came from a reputable journal
    - E.g., the person communicating with me is who they say they are

Replication/consensus etc. can help availability

- Confidentiality
  - My secret data was not leaked/stolen
    - E.g., the government, my spouse, a hacker did not gain access to my private information
- Integrity
  - Data / system was not tampered with
    - E.g., nobody added another ‘0’ in their bank account entry
    - E.g., nobody changed the content of my folder without me noticing
- **Availability**
  - Data or system will be available when needed
    - E.g., my website/service is protected from a malicious hacker trying to take them with a denial of service attack
- Authenticity
  - Belief in the source of the data
    - E.g., this new COVID study came from a reputable journal
    - E.g., the person communicating with me is who they say they are

Confidentiality: who is your adversary?

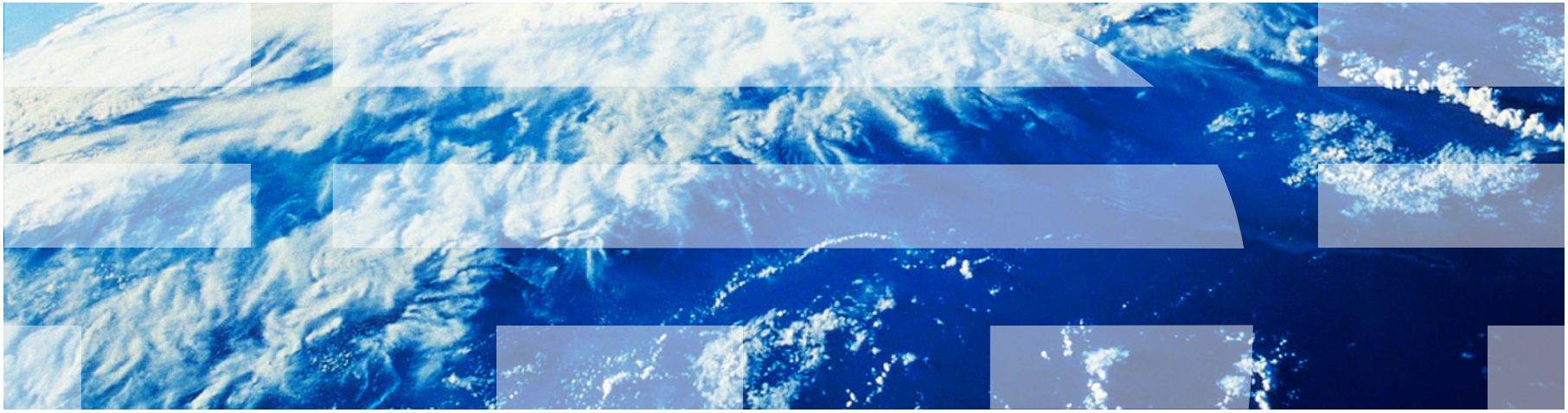
- Who are you afraid will access your data?
  - Hackers
  - Your husband/wife/boyfriend/girlfriend/mother/father…
  - Competitors
  - Anyone online
  - Your cloud provider (Google, Amazon, FB, etc.)
  - Your Internet provider
  - Your government
  - A foreign government
  - …
- → Need to explicitly define adversary

Goal: end-to-end security

- Confidentiality can be applied at any level of the stack
  - Data (e.g., encryption of data in a database, on a cloud file system)
  - Network (e.g., HTTPS)
  - Laptop/phone (e.g., full-disk encryption)
  - User identity (e.g., requiring password, multi-factor authentication)
  - …
- But applying security just at a single level does not mean your data is secure
- For example:
  - Uploading my password to a phishing site over HTTPS
  - Thief steals my phone and phone PIN, can access my phone's disk-encrypted data

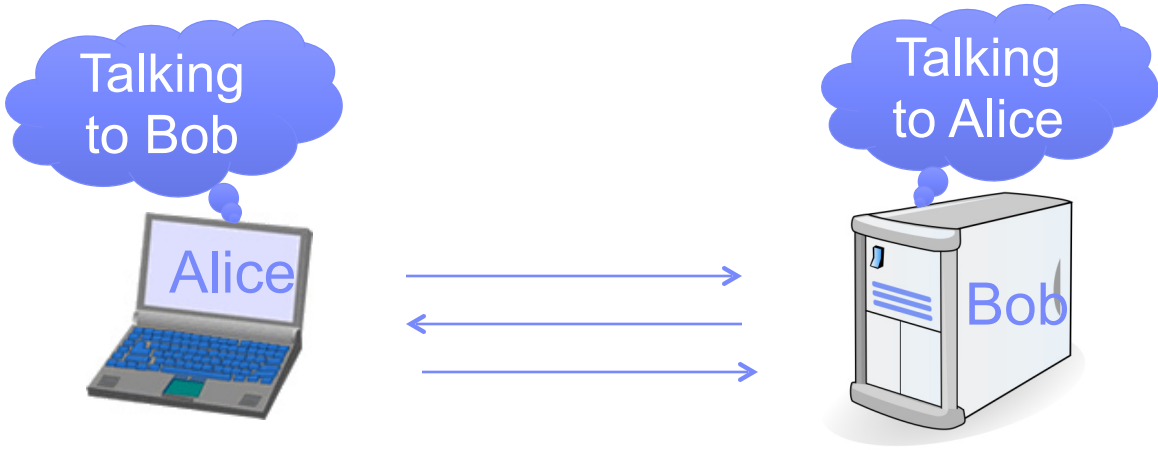
Some slides borrowed from Dan Boneh

Cryptography

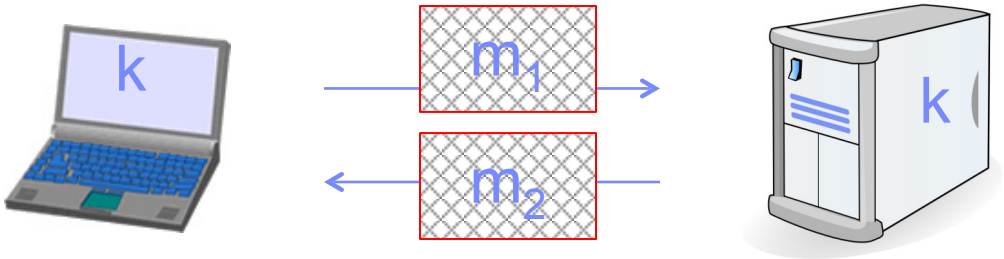


Crypto core

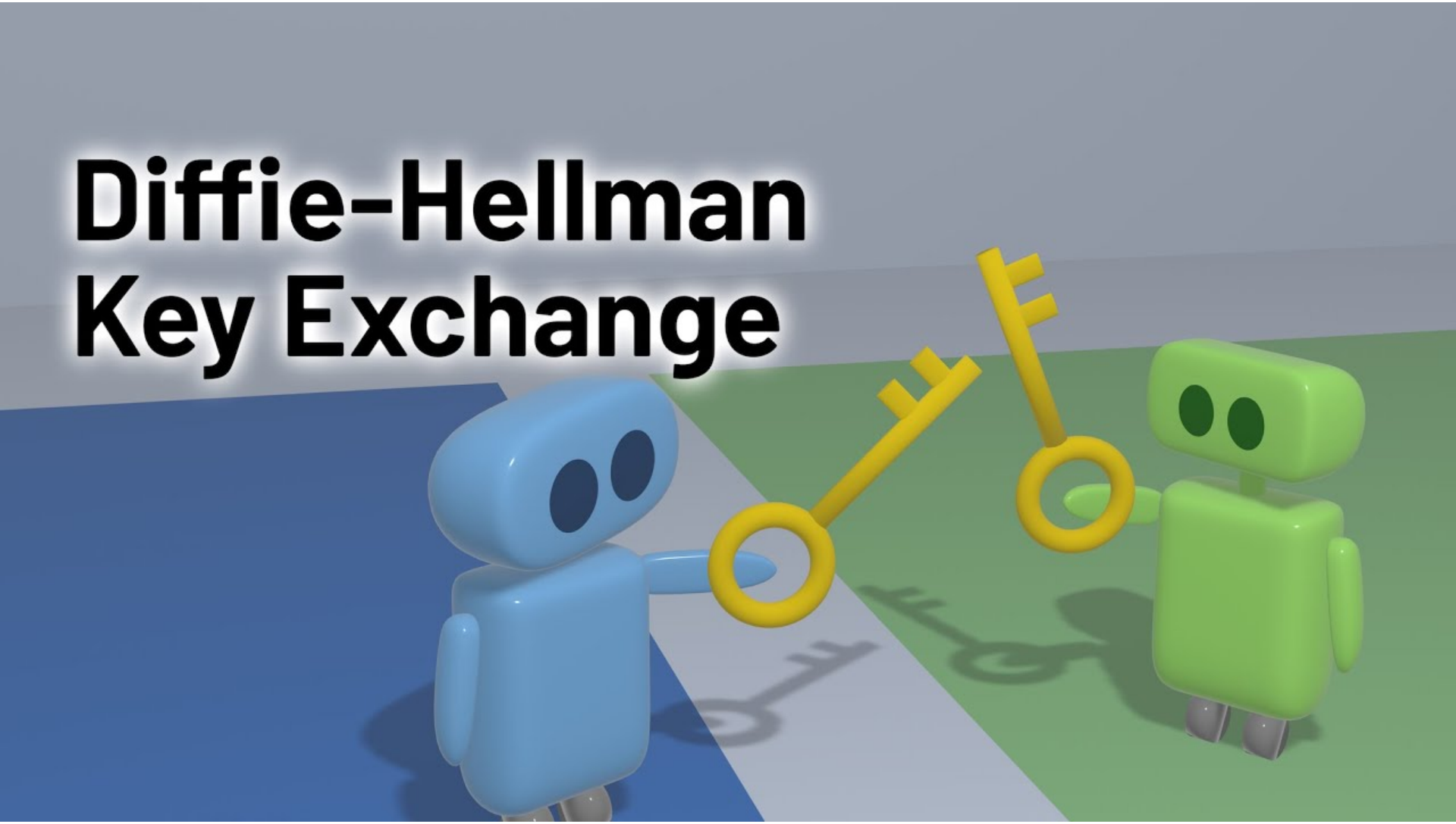
Secret key establishment:



Secure communication:



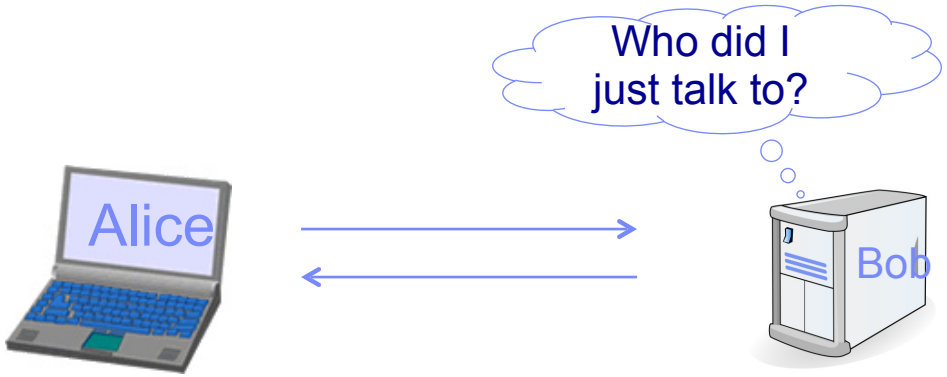
confidentiality and integrity


Diffie-Hellman
Key Exchange

But crypto can do much more

- Digital signatures (authentication)

- Anonymous communication



- Cryptocurrencies and blockchain

Symmetric encryption

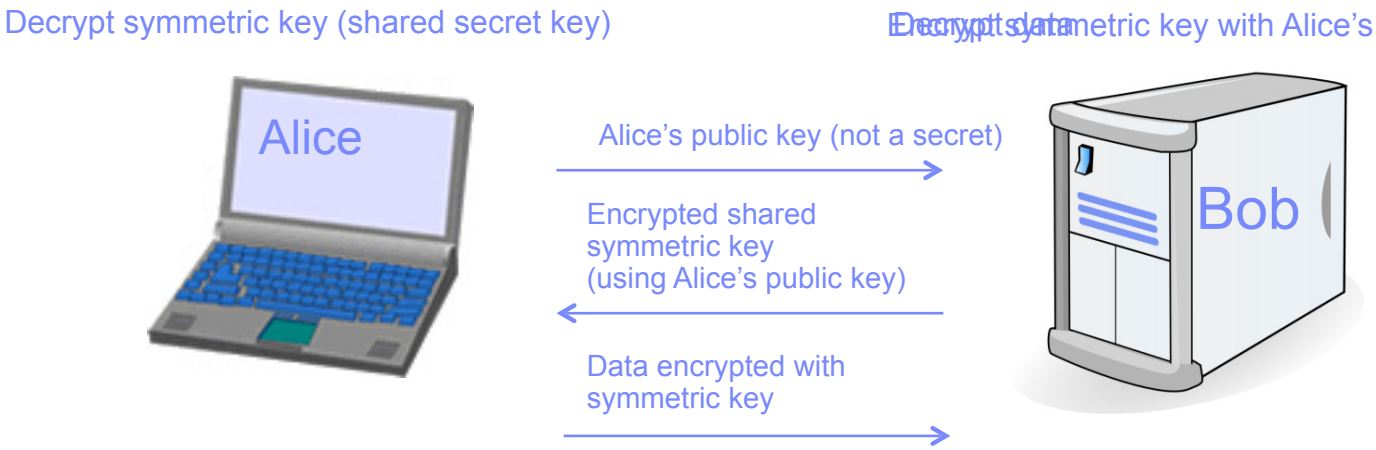
- Encryption function E(x,k)=y produces an output that appears to be totally random
  - x is unencrypted data (plaintext), k is key
- Decryption function D(y,k)=x
  - Decryption function is usually much harder to compute than encryption function
- The key is secret
  - If you obtain the key, you can decrypt the data
- Recall that hashing function h(x) produces a random output
  - Hashing is like a one-way encryption (can't be decrypted)
  - Hashing outputs can be the same for different inputs if the output space of the function is small (e.g., Bloom filter false positives)
    - In cryptography we try to avoid that

Symmetric encryption in the real world

- Symmetric encryption is the workhorse of encryption techniques
  - Used to encrypt/decrypt bulk data (storage, network packets, etc.)
- Most common algorithm: AES
  - Use a single secret key to encrypt and decrypt
  - Usually relatively good performance
  - Modern CPUs have support for high AES throughput
- Most modern algorithms are heuristic based
  - There is no formal proof that AES decryption is hard
  - But in practice it has withstood many attempts to hack over the years

Asymmetric encryption

- Used for confidentiality, integrity and authenticity
- Similar to symmetric encryption, except encryption and decryption use different keys
- Encryption algorithm uses **public key**, which is published (not a secret)
- Decryption algorithm uses **private key**, which is a secret
- Why is it useful to use different keys for encryption and decryption?
  - For example, if two parties want to share a secret (e.g., a symmetric key) over untrusted network

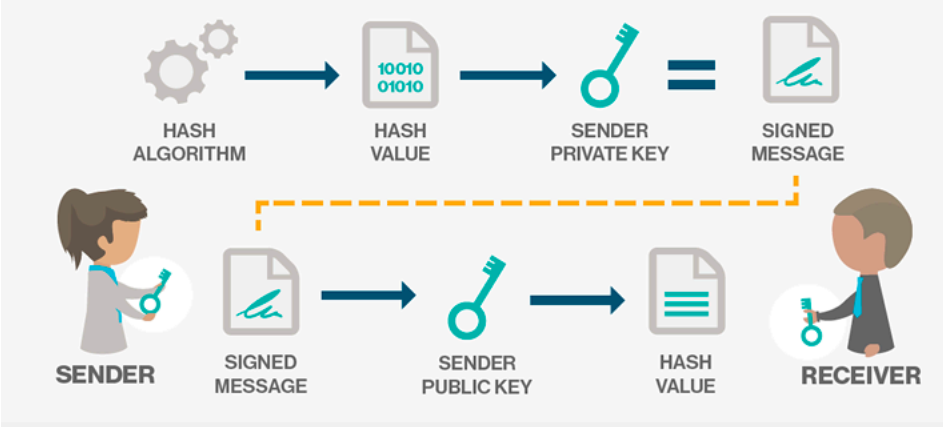


Asymmetric encryption in the real world

- Asymmetric encryption is mostly used to encrypt symmetric keys or small shared secrets
  - Why? Because it usually has lower performance than symmetric encryption
- Most common algorithms: RSA
- RSA is more theoretically sound than AES
  - Based on simple yet ingenious idea that multiplication is much easier than factorization
- Basic idea:
  - Private key is based on two very large prime numbers
  - The public key is their product
- Even so, factorization problem has not been proven as “hard”

Digital signatures

- Used for integrity and authenticity (not confidentiality)
- Goal: anyone can verify the person sending the message has the private key (the secret)
- Step 1: hash your data
  - Also called a digest
- Step 2: use private key to sign message
- Append signature to your message
- Receiver can then verify using the sender's public key

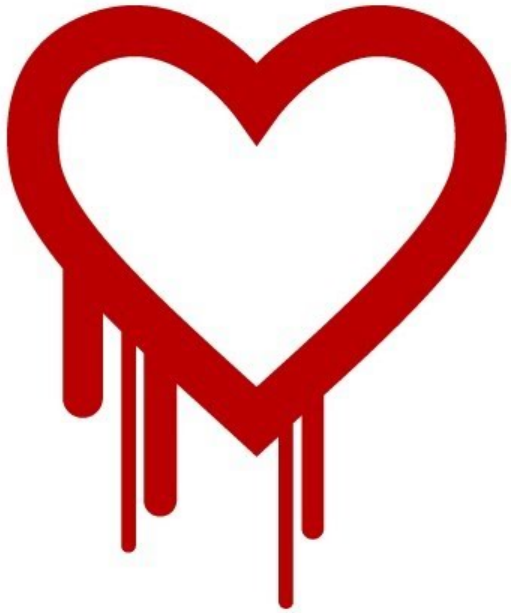


Digital signatures in the real world

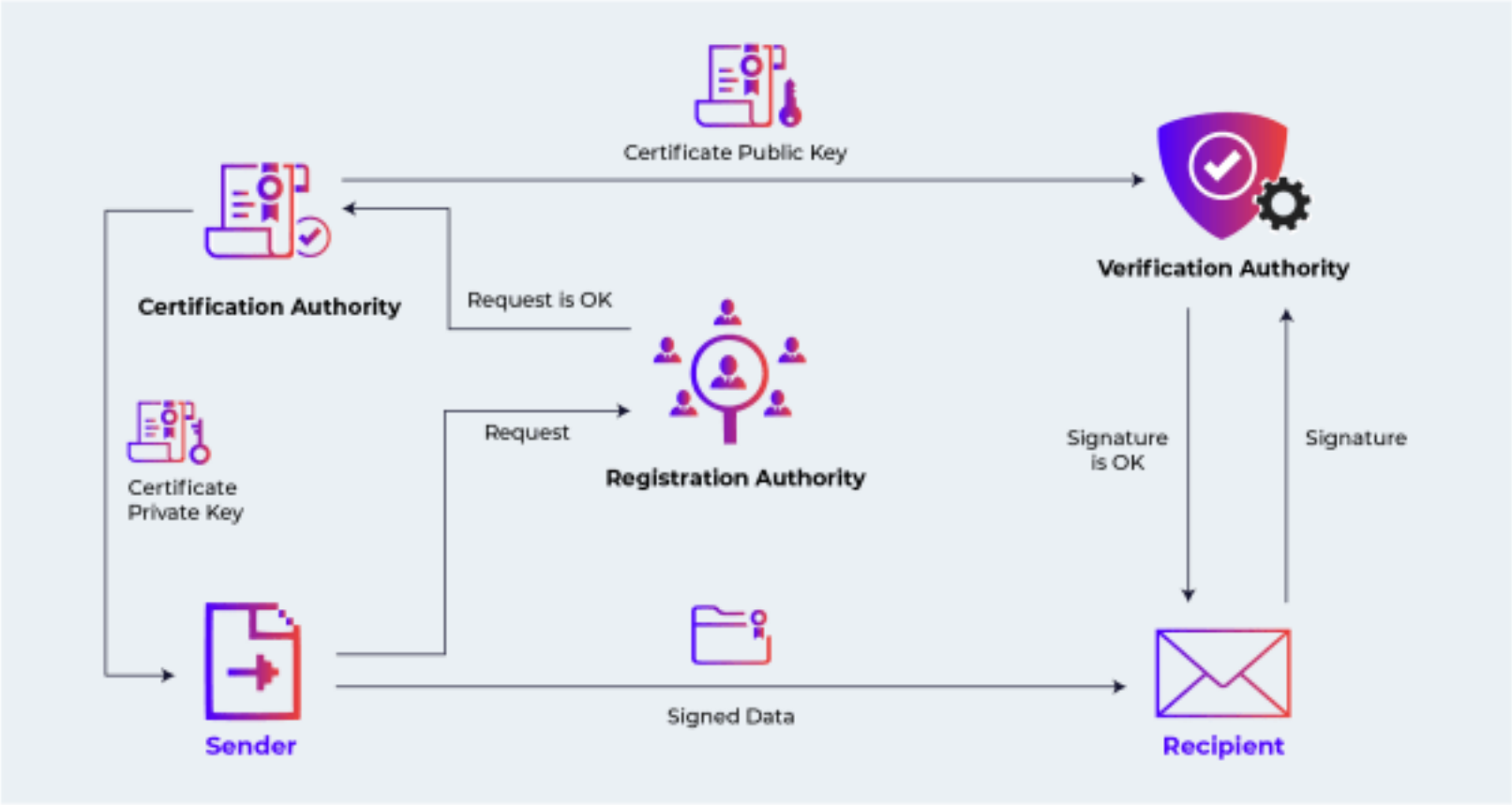
- Used widely for authentication
  - This website actually belongs to google.com
  - This email was sent from columbia.edu
  - This credit card is real
- Common algorithms: DSA, RSA signatures
  - Based on asymmetric encryption
  - Generally slow

Encryption in the real world

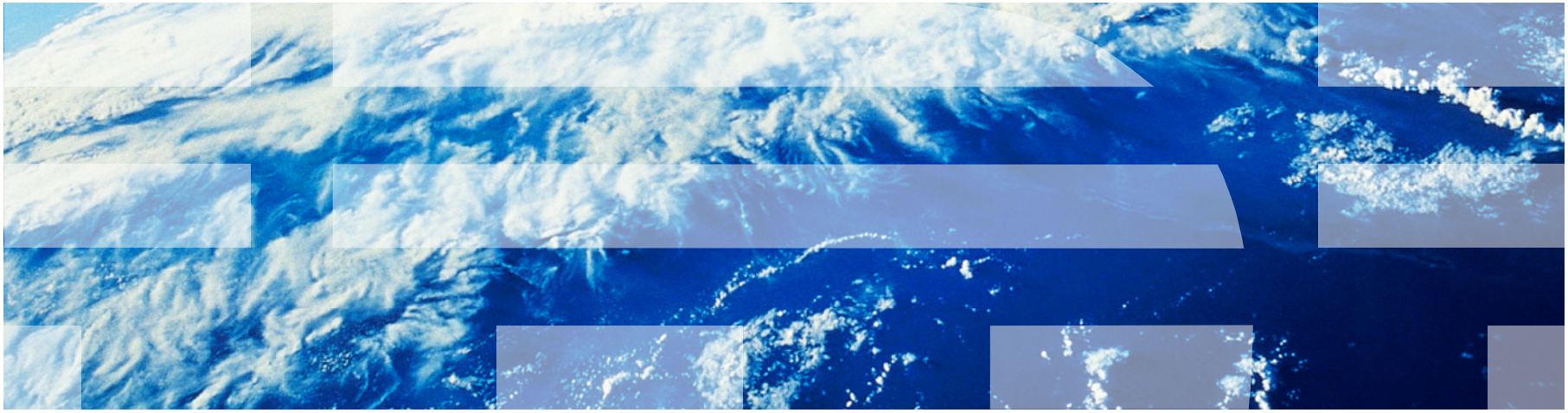
- You should never invent your own encryption algorithm or protocols
  - Like every other rule: “unless you know what you’re doing”
- Always use existing libraries / protocols
  - Even when using existing encryption libraries, it is tricky not to use them incorrectly
- Even widely used open source protocols have had security flaws
  - E.g., OpenSSL Heartbleed vulnerability



Public Key Infrastructure



Data Compliance and Privacy



Security and compliance are not the same

- Data compliance definition:
  - The process of ensuring that a dataset confirms to the rules specified by national or international laws, or the standards set by trade bodies
- Sometimes compliance includes security measures, but not always
- Compliance is primarily a legal framework on how to handle data
  - Usually more concerned with privacy than security
- Being compliant does not mean your data is secure!

Types of Data Compliance

- HIPAA
  - Ensuring the privacy of healthcare patient data
- GDPR
  - Ensuring privacy of citizens of the European Union
- PCI
  - Protecting credit card data
- FERPA
  - Protecting privacy of student information (personal details, grades, etc.)
- …

Does compliance affect you?

- Compliance affects the following use cases:
  - Dealing with personally identifiable information (PII), i.e., personal details of humans (names, locations, email addresses, phone numbers, social security numbers, etc.)
  - Healthcare
  - Finance
  - Education
  - Defense
- Usually relevant if you work on some kind of private data

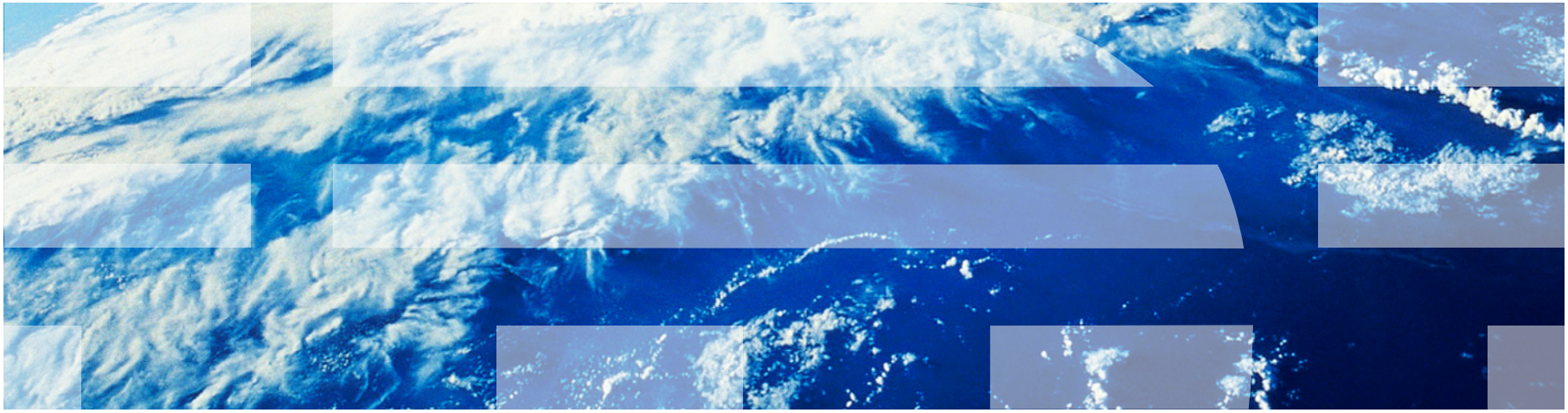
General principles

- “Protect” PII
  - This usually means some kind of encryption, but it’s not always clear at what level
- Auditability
  - Need a permanent log of any operation on the PII
    - Who read/modified/deleted, when and what did they do
  - Cannot delete log
- Access control
  - Restrictions on who can access PII
- Restrictions on sharing PII
  - Often needs to be shared securely
  - Might require the other side to sign a legal agreement
- “Right to be forgotten”
  - If user asks to delete their data, you need to do so
- Restrictions on where data can be stored
  - E.g., EU PII can only be stored in EU

How do big data systems implement compliance?

- Certain systems might be designated for storing PII, separate from other non-sensitive systems
- Require access control (who can access/update system, be able to revoke access)
- Require audit logging
- Encryption of sensitive data within the system (**encryption at rest**) and in transit when ingesting/leaving to and from the system (**encryption in transit**)
- Often separate data from different regions
  - E.g., EU storage system sits in European datacenter
- The major public cloud providers have built-in compliance controls

Security and Compliance Tips and Tricks



Principles for keeping your sensitive data confidential

- Always use two-factor authentication!
  - If your password is stolen, you're still ok
  - Best is hardware token, but other forms (text/phone call) are ok
- Use password manager if possible
  - Chrome has a pretty useful built-in one
- Never reuse a password for an important system
  - E.g., Columbia account, email, bank, server…
- If you work at a company, implement single-sign on system
  - E.g., Auth0, Okta, Onelogin…
- If you work at a company, hire 3rd party auditing/penetration testing
  - They will always find something you didn't think about
- Email/text is the most common vector of attack: be careful of suspicious links or fake emails from your boss/professor/colleague
- Never design your own cryptographic protocol
- Compliance != security
  - You can be HIPAA/GDPR/PCI compliant but completely insecure

Principles for remaining compliant

- Never store user passwords, credit cards, anywhere on your system
  - Always better to outsource payments to third-party service (e.g., Stripe, Plaid…)
- Understand which compliance regulation you need to follow
  - Most common: PCI/HIPAA/GDPR
- Create a document with clear guidelines
  - E.g., Users' names and addresses are only stored here, this application can only access this database, etc.
- Consult with a lawyer
  - But don't forget, lawyers don't understand security ☺
- Get third-party auditing/certification when possible (not always possible)