Blockchain Theory 1

Week 1
Lesson 1

Lesson Plan

Decentralised Systems
Blockchain Components (simplified)
Cryptography and blockchain history

Decentralised Systems

Problems with centralised systems

Monetary System

- Bank closure / insufficient capital reserves
    - greek debt crisis in 2015 ? banks closed and people lost savings, insurance schemes meant nothing, lead to an increase in Bitcoin use in Greece
- Availability of banks
- Inflation - money supply controlled by central authority
- Merchant accounts may be shut down
- Control of money for political reasons - wikileaks funding shutdown

There are layers of access control built into our banking systems to prevent fraudulent transactions, effectively security is achieved by closing the network.

See The Internet of Money - Andreas M. Antonopoulos

Goals of decentralisation

- Participation
- Diversity
- Efficiency
- Conflict resolution
- Flexibility
- Moving power to the edge (user)

◯ Decentralized	△ Centralized
Inefficient	Efficient
Dangerous	Safe
No built-in trust framework	Trust through structure
Anarchy	Absolute Power / Absolute Corruption
Distributed Wealth	Consolidated Wealth
Generalized	Specialized
Flexible	Structured
Resilient	Fragile
Personal Sovereignty	Ceded Authority
Complex to manage	Simple to manage
Difficult to scale	Easy to scale
Local Overheads	Global Overheads

From

Betakit Blog

Class Exercise

Imagine playing Monopoly over the internet (using the physical game, without a software version)
(Dice , board position, account balance, hotels possessed etc.)

What are the problems ?

How can we solve these ?

A prototype cryptocurrency ?

Rai Stones

The monetary system of Yap relies on an oral history of ownership. In the case of stones that are too large to move, buying an item with one simply involves agreeing that the ownership has changed. As long as the transaction is recorded in the oral history, it will now be owned by the person to whom it is passed and no physical movement of the stone is required.

Components of a blockchain

Gossip Network

Shared Public Ledger

Cryptography

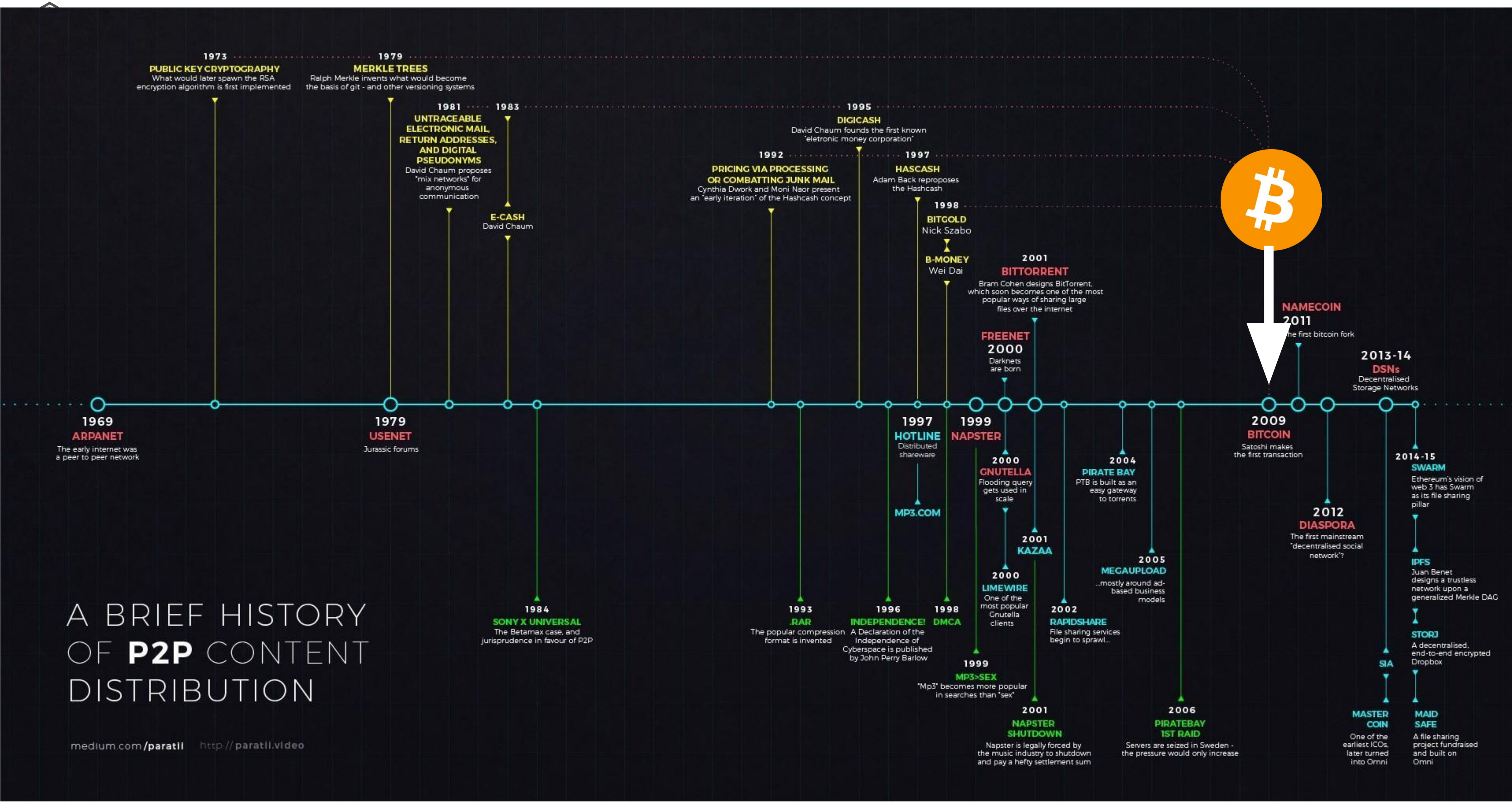
Components of the Blockchain

- Shared public ledger updated by consensus
- P2P Network
- Cryptography

These components give a blockchain system

- Transparency and verifiable state based on consensus
- Resilience
- Censorship resistance
- Tamper proof interactions

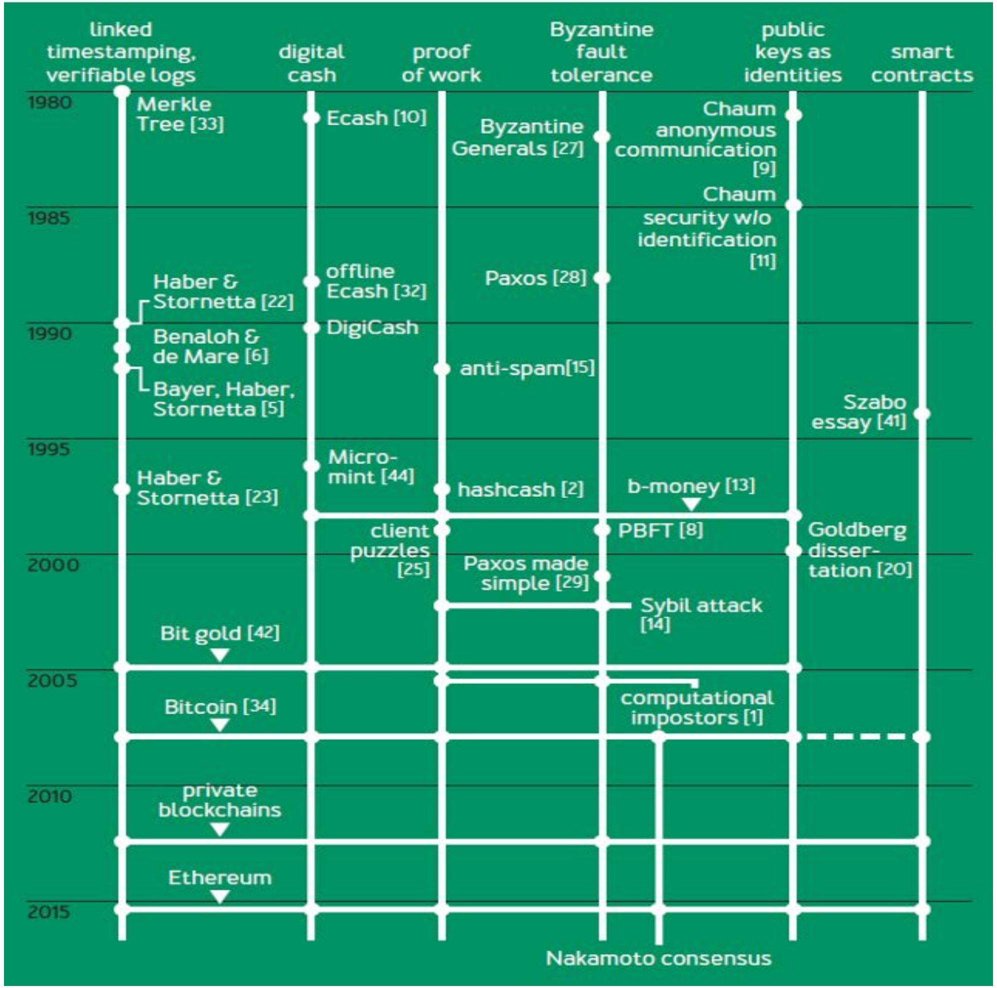
Timeline of cryptographic systems


A BRIEF HISTORY OF P2P CONTENT DISTRIBUTION
medium.com/paratii
http://paratii.video
1969
ARPANET
The early internet was a peer to peer network
1973
PUBLIC KEY CRYPTOGRAPHY
What would later spawn the RSA encryption algorithm is first implemented
1979
MERKLE TREES
Ralph Merkle invents what would become the basis of git - and other versioning systems
1979
USENET
Jurassic forums
1981
UNTRACEABLE ELECTRONIC MAIL, RETURN ADDRESSES, AND DIGITAL PSEUDONYMS
David Chaum proposes "mix networks" for anonymous communication
1983
E-CASH
David Chaum
1984
SONY X UNIVERSAL
The Betamax case, and jurisprudence in favour of P2P
1992
PRICING VIA PROCESSING OR COMBATTING JUNK MAIL
Cynthia Dwork and Moni Naor present an "early iteration" of the Hashcash concept
1993
.RAR
The popular compression format is invented
1995
DIGICASH
David Chaum founds the first known "eletronic money corporation"
1996
INDEPENDENCE!
A Declaration of the Independence of Cyberspace is published by John Perry Barlow
1997
HASCASH
Adam Back reproposes the Hashcash
1997
HOTLINE
Distributed shareware
MP3.COM
1998
BITGOLD
Nick Szabo
B-MONEY
Wei Dai
1998
DMCA
1999
NAPSTER
1999
MP3>SEX
"Mp3" becomes more popular in searches than "sex"
FREENET
2000
Darknets are born
2000
GNUTELLA
Flooding query gets used in scale
2000
LIMEWIRE
One of the most popular Gnutella clients
2001
BITTORRENT
Bram Cohen designs BitTorrent, which soon becomes one of the most popular ways of sharing large files over the internet
2001
KAZAA
2001
NAPSTER SHUTDOWN
Napster is legally forced by the music industry to shutdown and pay a hefty settlement sum
2002
RAPIDSHARE
File sharing services begin to sprawl...
2004
PIRATE BAY
PTB is built as an easy gateway to torrents
2005
MEGAUPLOAD
...mostly around ad-based business models
2006
PIRATEBAY 1ST RAID
Servers are seized in Sweden - the pressure would only increase
2009
BITCOIN
Satoshi makes the first transaction
NAMECOIN
2011
The first bitcoin fork
2012
DIASPORA
The first mainstream "decentralised social network"?
2013-14
DSNs
Decentralised Storage Networks
SIA
MASTER COIN
One of the earliest ICOs, later turned into Omni
2014-15
SWARM
Ethereum's vision of web 3 has Swarm as its file sharing pillar
IPFS
Juan Benet designs a trustless network upon a generalized Merkle DAG
STORJ
A decentralised, end-to-end encrypted Dropbox
MAID SAFE
A file sharing project fundraised and built on Omni

Working Towards Bitcoin

- Timestamping and verification
- Digital Cash
- Proof of Work (Consensus)
- Byzantine Fault Tolerance (Consensus)
- Public keys as Identities
- Smart Contracts

FIGURE 1: CHRONOLOGY OF KEY IDEAS FOUND IN BITCOIN



1970s

The Early days of internet

(1969)
The ARPANET (early Internet) was a p2p network

Problem = Security !

1) Privacy

- How do I ensure that my message has not been modified ?

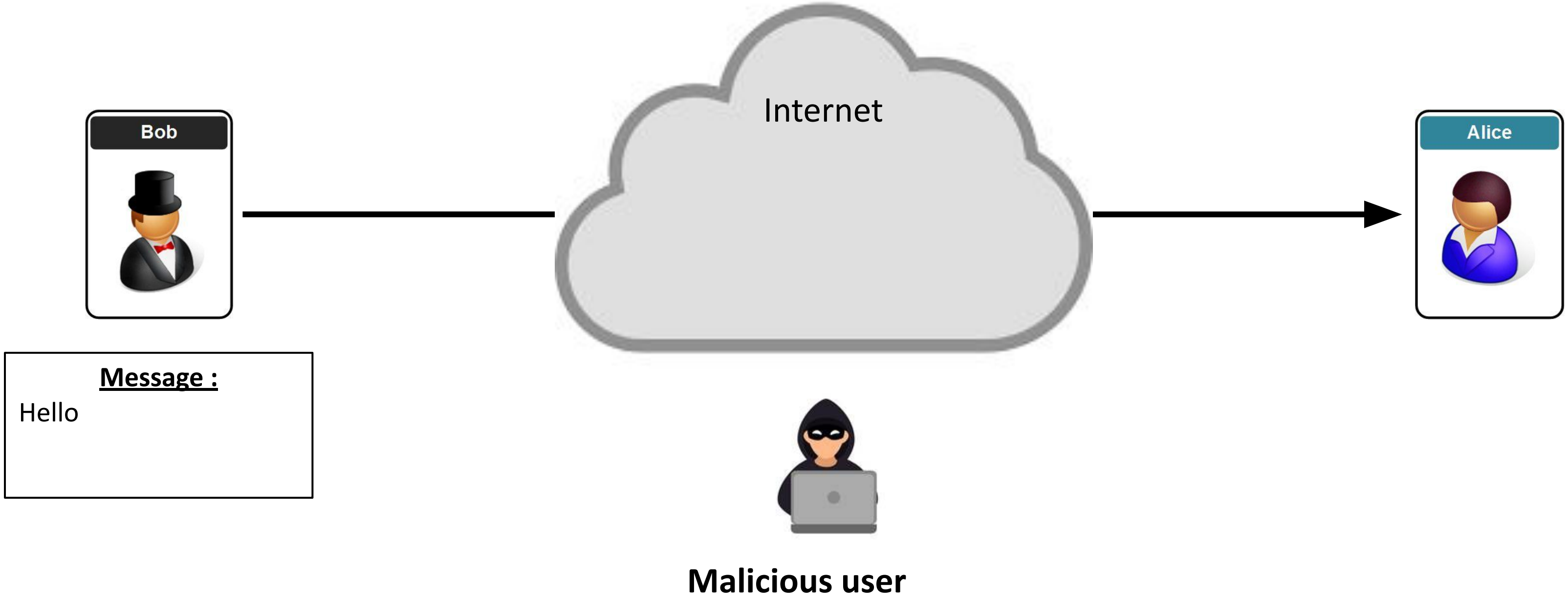
2) Authenticity

- How do I ensure that the message comes from a legitimate person ?

1970's

Secure Communication over Insecure Channel

<u>Problem 1 :</u> How do I ensure that my message has not been modified ?



1970's

Secure Communication over Insecure Channel

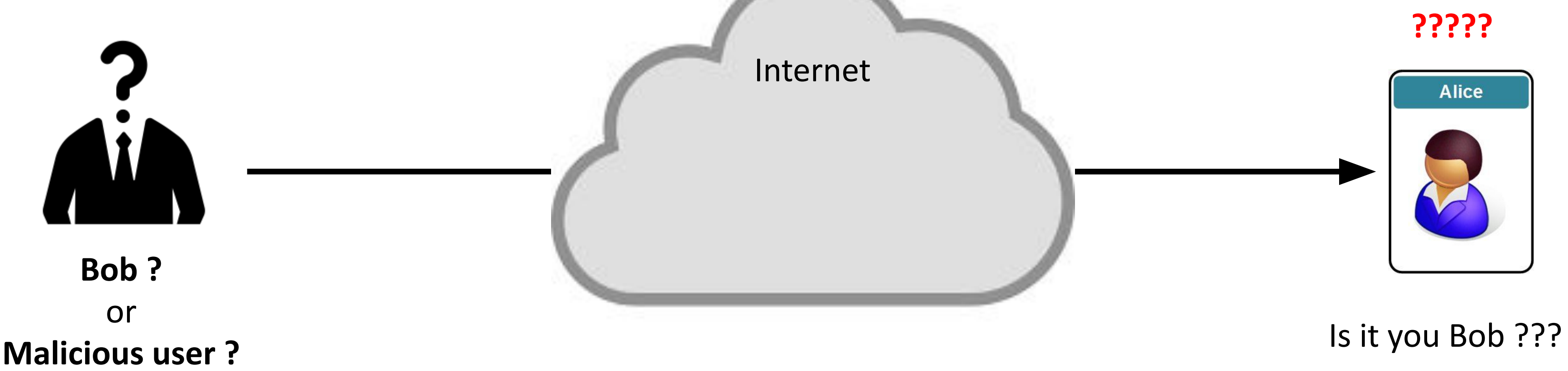
<u>Problem 1 :</u> How do I ensure that my message has not been modified ?

Malicious user

1970's

Secure Communication over Insecure Channel

Problem 2 : How do I ensure that the message comes from a legitimate person ?



1970’s

(1969)
The ARPANET (early Internet) was a p2p network

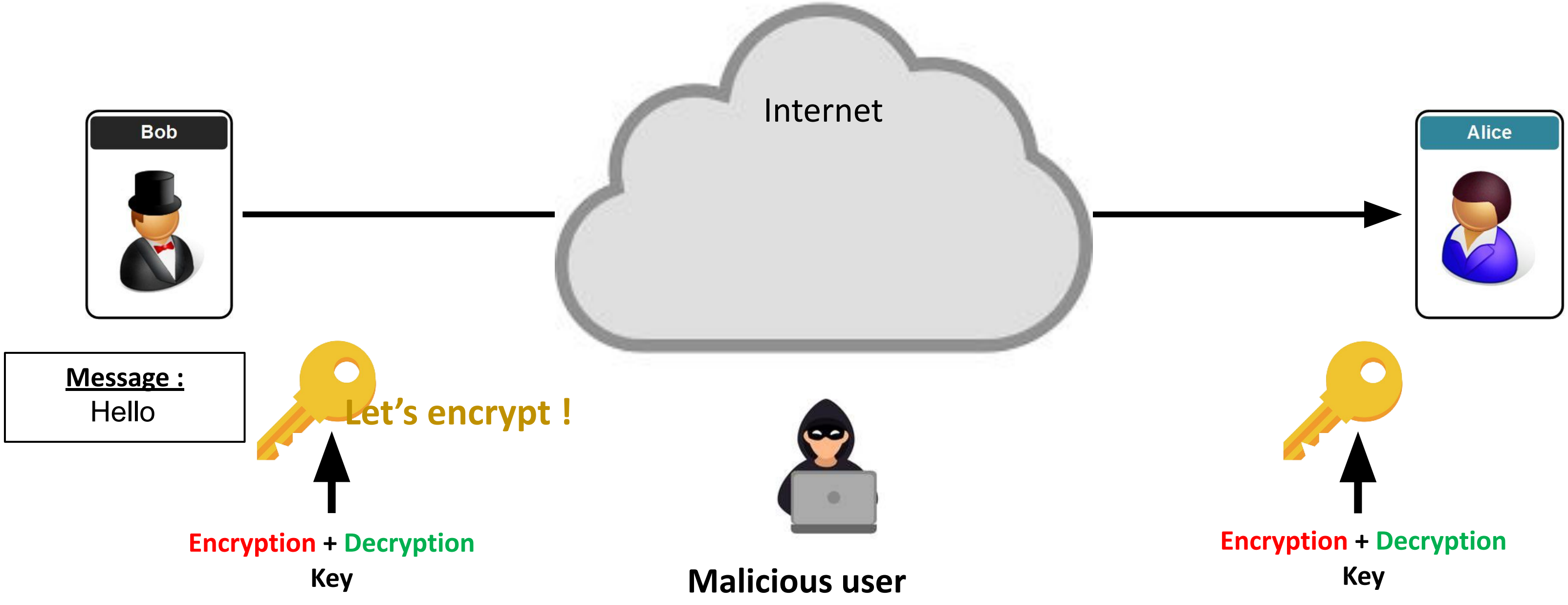
1st Solution : Symmetric Cryptography !

- Alice and Bob **share the same key**.
- One key for **both encryption** and **decryption** of messages

1970’s

Secure Communication over Insecure Channel

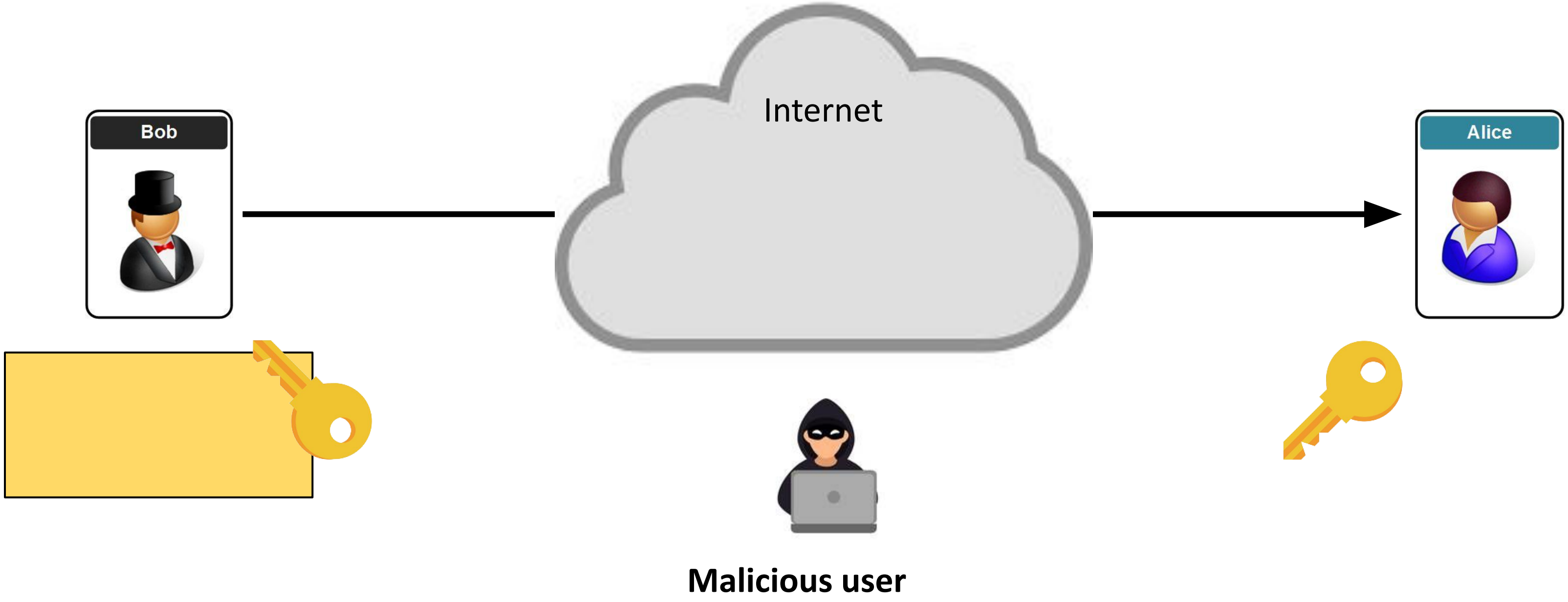
Symmetric Cryptography : **encryption and decryption keys are the same**



1970's

Secure Communication over Insecure Channel

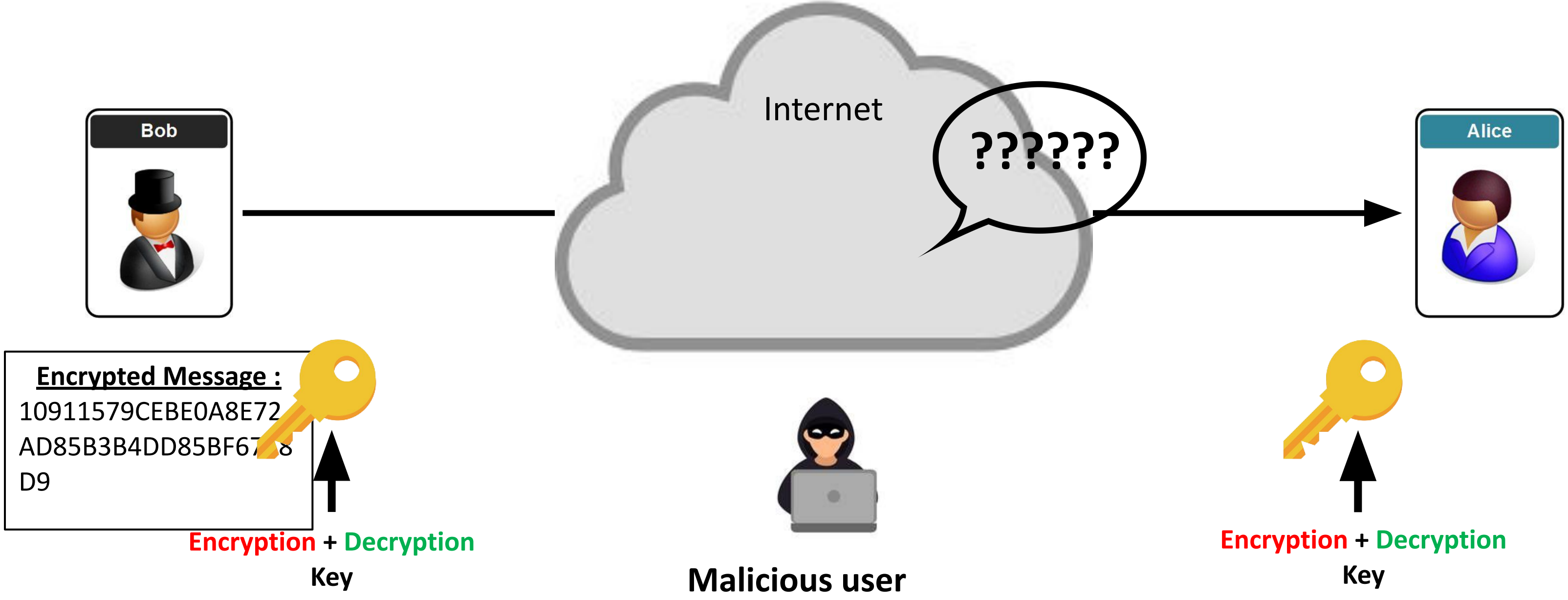
<u>Symetric Cryptography :</u> encryption and decryption keys are the same



1970’s

Secure Communication over Insecure Channel

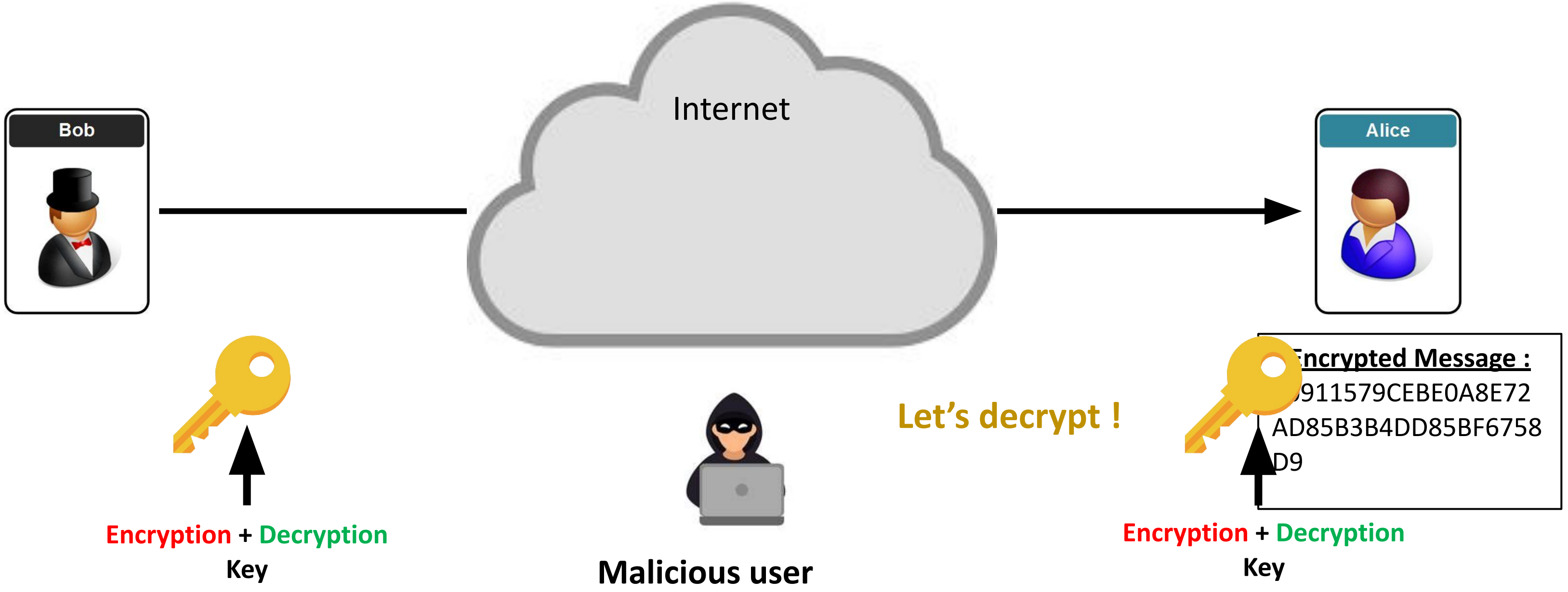
<u>Symetric Cryptography :</u> encryption and decryption keys are the same



1970’s

Secure Communication over Insecure Channel

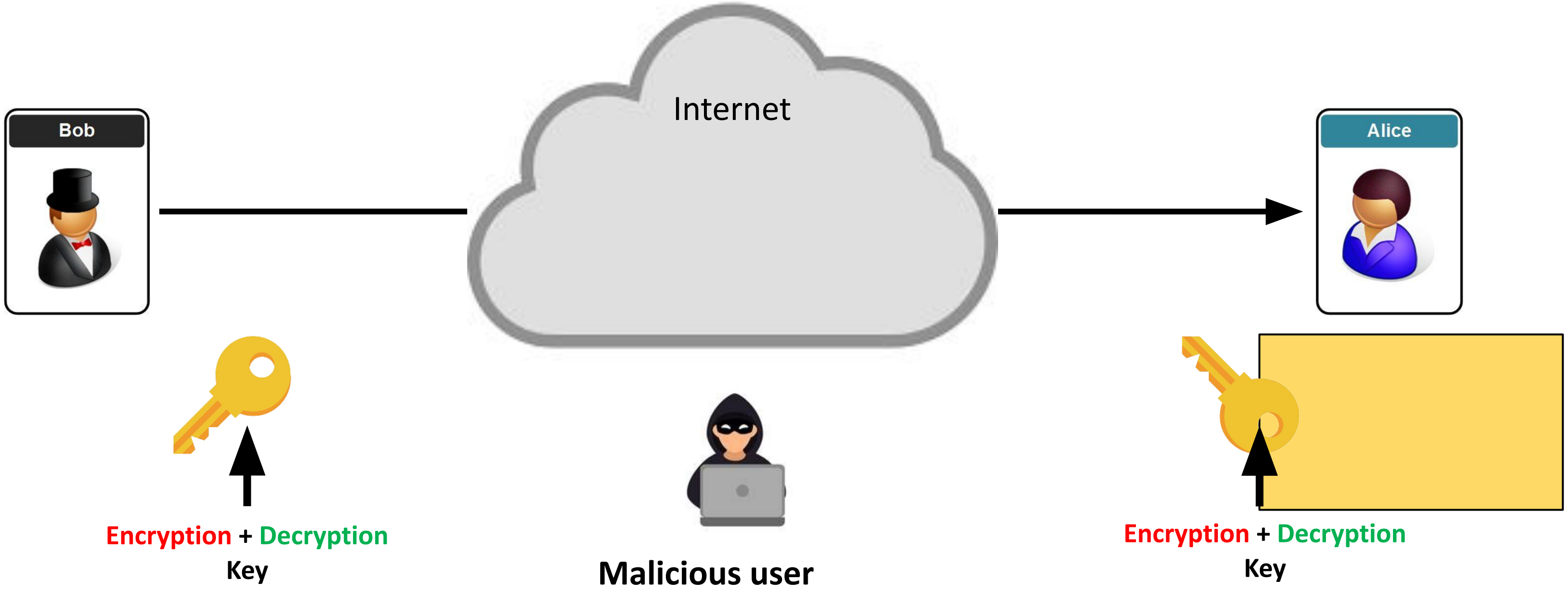
Symetric Cryptography : encryption and decryption keys are the same



1970's

Secure Communication over Insecure Channel

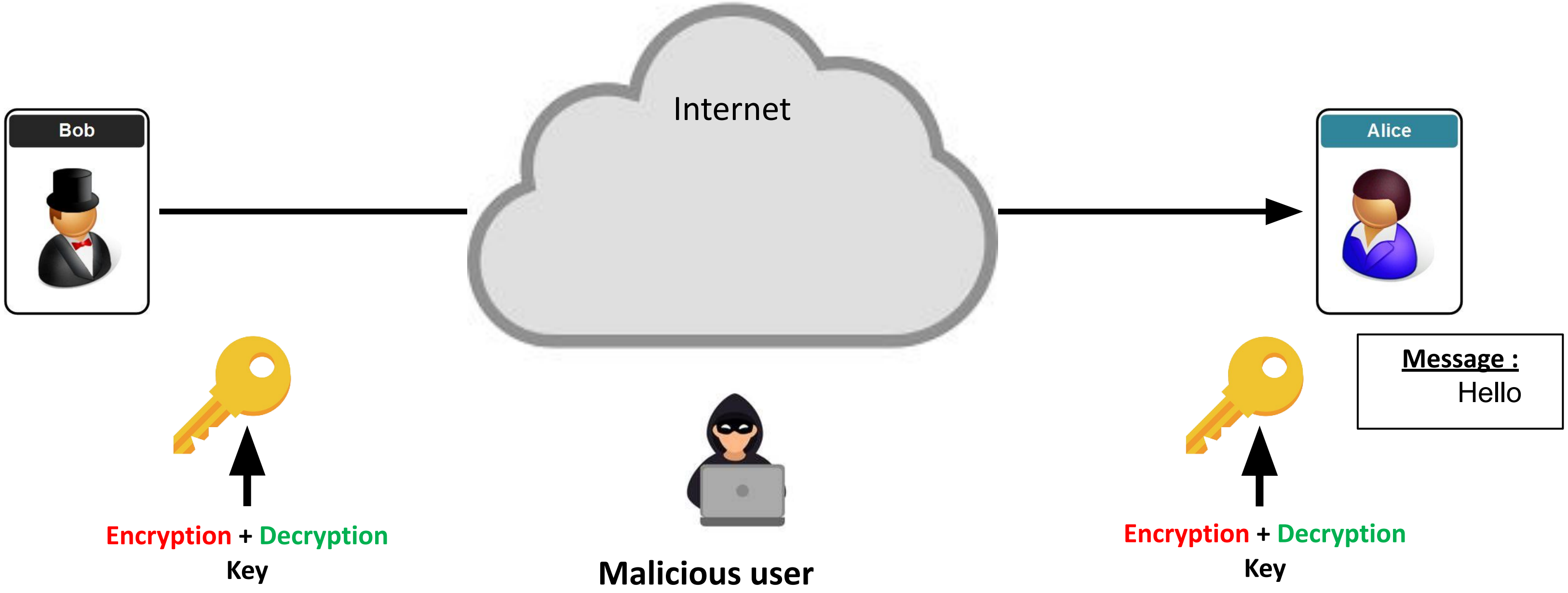
<u>Symetric Cryptography :</u> encryption and decryption keys are the same



1970's

Secure Communication over Insecure Channel

Symetric Cryptography : **encryption** **and** **decryption** **keys are the same**



But what about key management ?

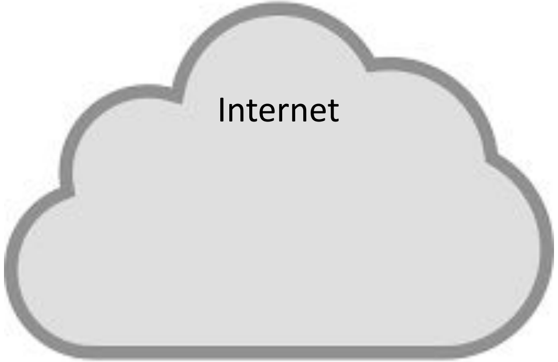
Can Alice and Bob share a key

- Without meeting
- Across a potentially hostile network

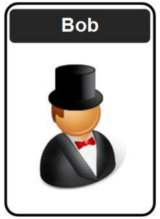
EXTROPY.IO

Public

Private



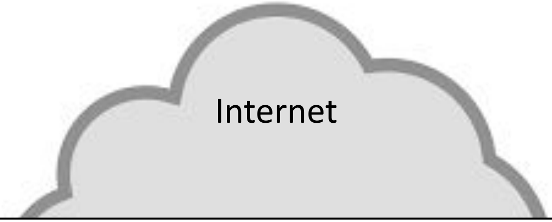
Private

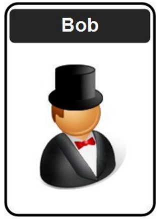


Private

Public

Private





Step 1 :

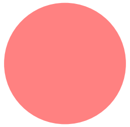
Alice and Bob :

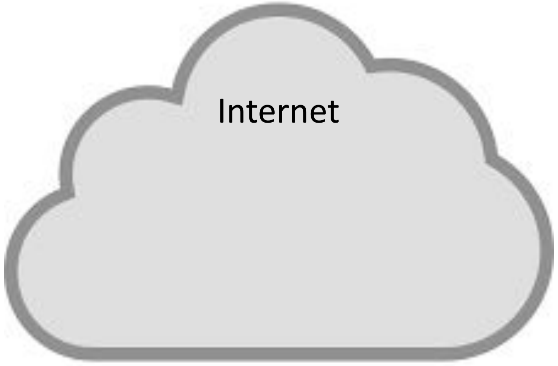
Select a random secret individually !

Public

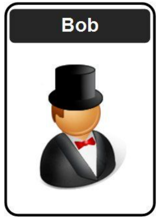
Private

Private Key A





Private

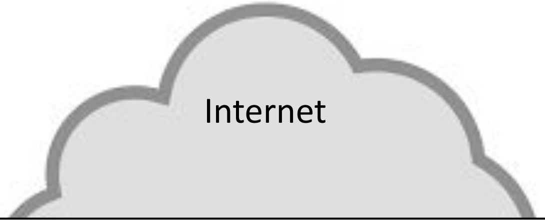


Private Key B

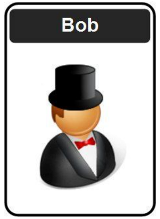
Private

Private Key A

Public



Private



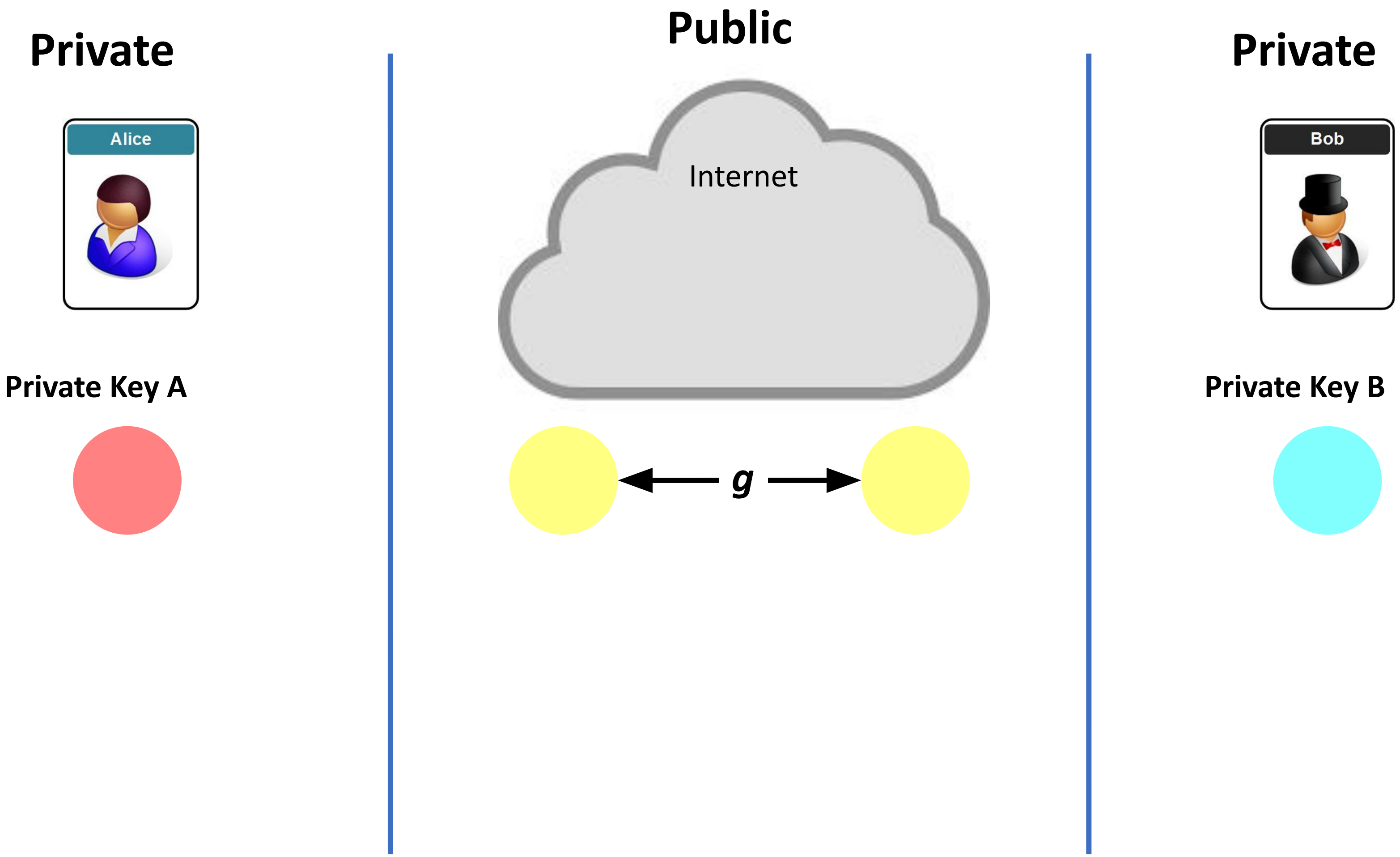
Private Key B

Step 2 :

Alice and Bob :

exchange **a common value** (*g)* on the internet !

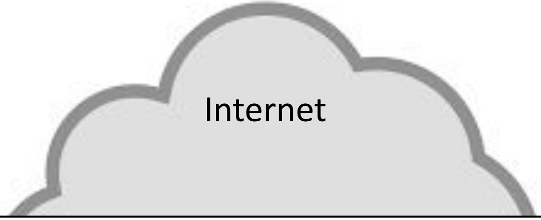
(Insecure network)


Private
Public
Private
Internet
Alice
Bob
Private Key A
Private Key B
g

Private

Private Key A

Public



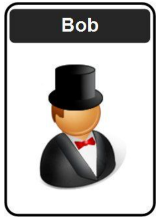
Step 3 :

Alice and Bob

combine their private key with the common value *g* !

(in their private network)

Private



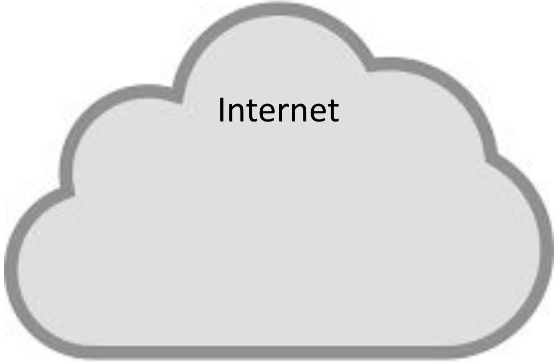
Private Key B

Private

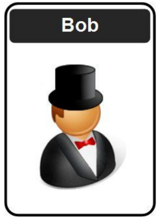
Private Key A

A (g)

Public



Private



Private Key B

B (g)

Private

Public

Private

Alice

Internet

Bob

Step 4 :

Private Key A

Private Key B

Alice and Bob

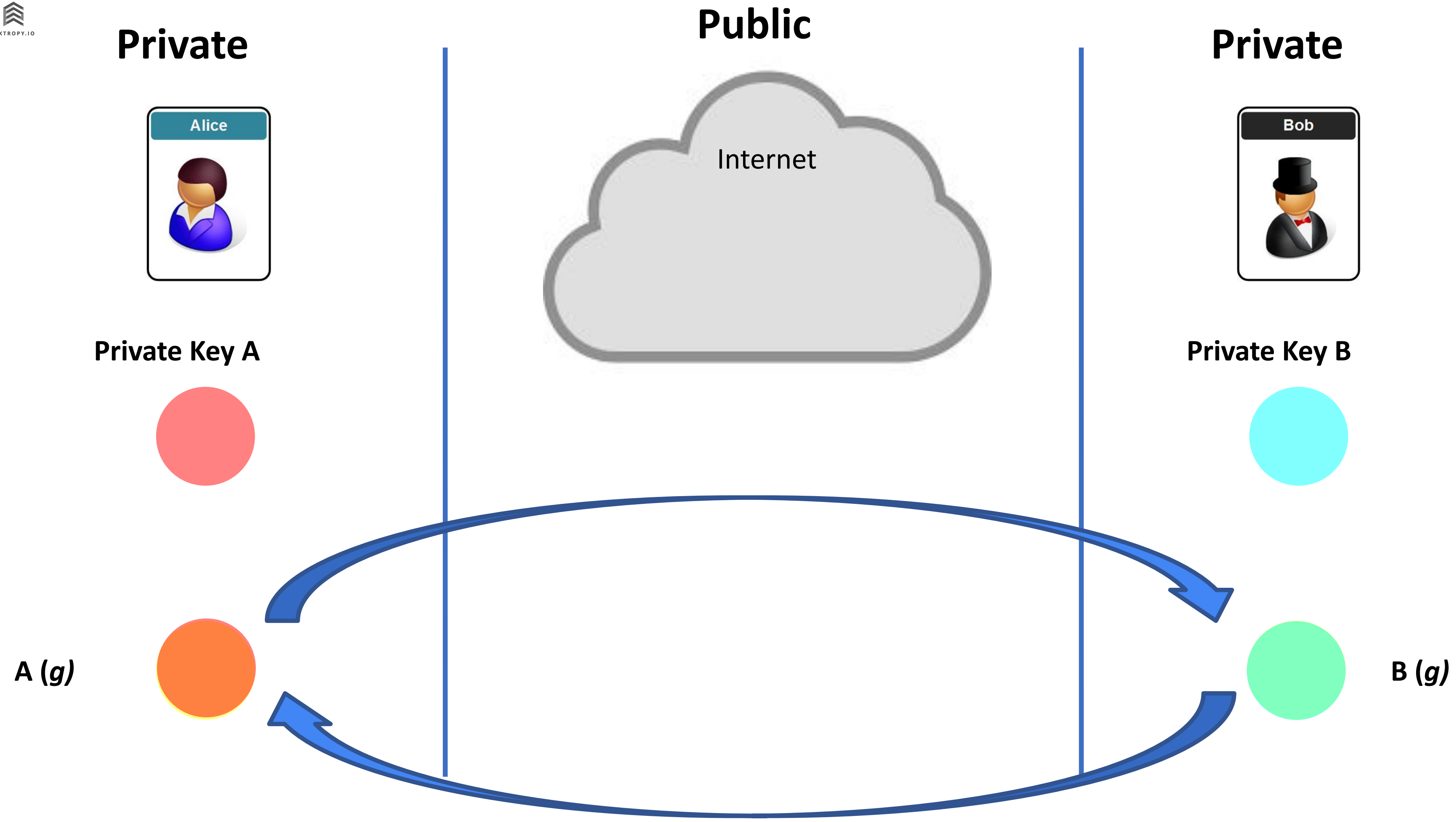
Exchange both their combine value

A (*g*) and B (*g*)

(over the internet)

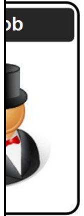
A (*g*)

B (*g*)



Private
Public
Private
Alice
Internet
Bob
Private Key A
Private Key B
A (g)
B (g)

Private **Public** **Private**



<u>Final Step !</u>

Alice and Bob

Combine their private key with the combine key of the other to obtain a shared secret key

PK(A) + B (*g*)

PK(B) + A (*g*)

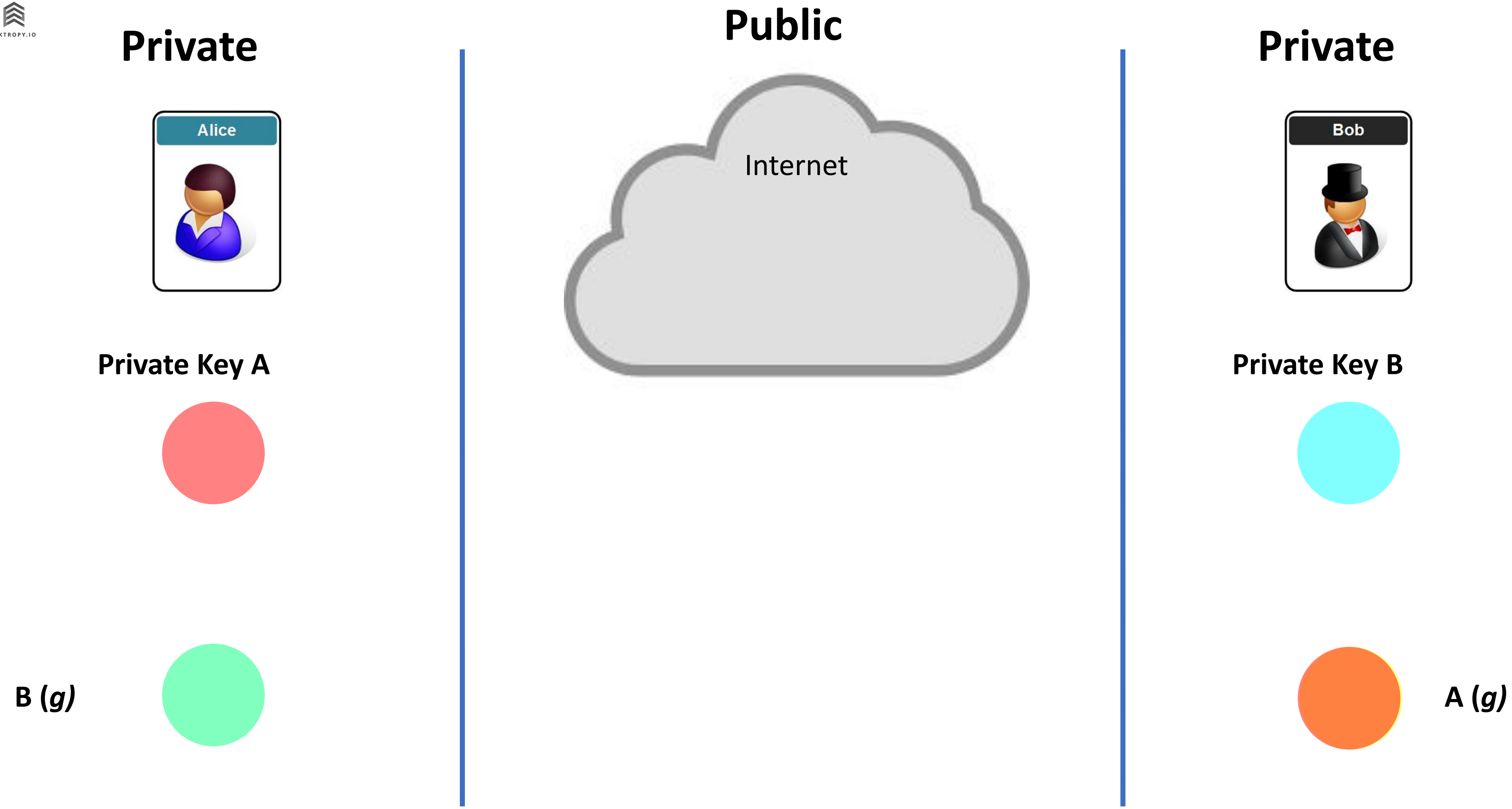
(privately)

Private Key

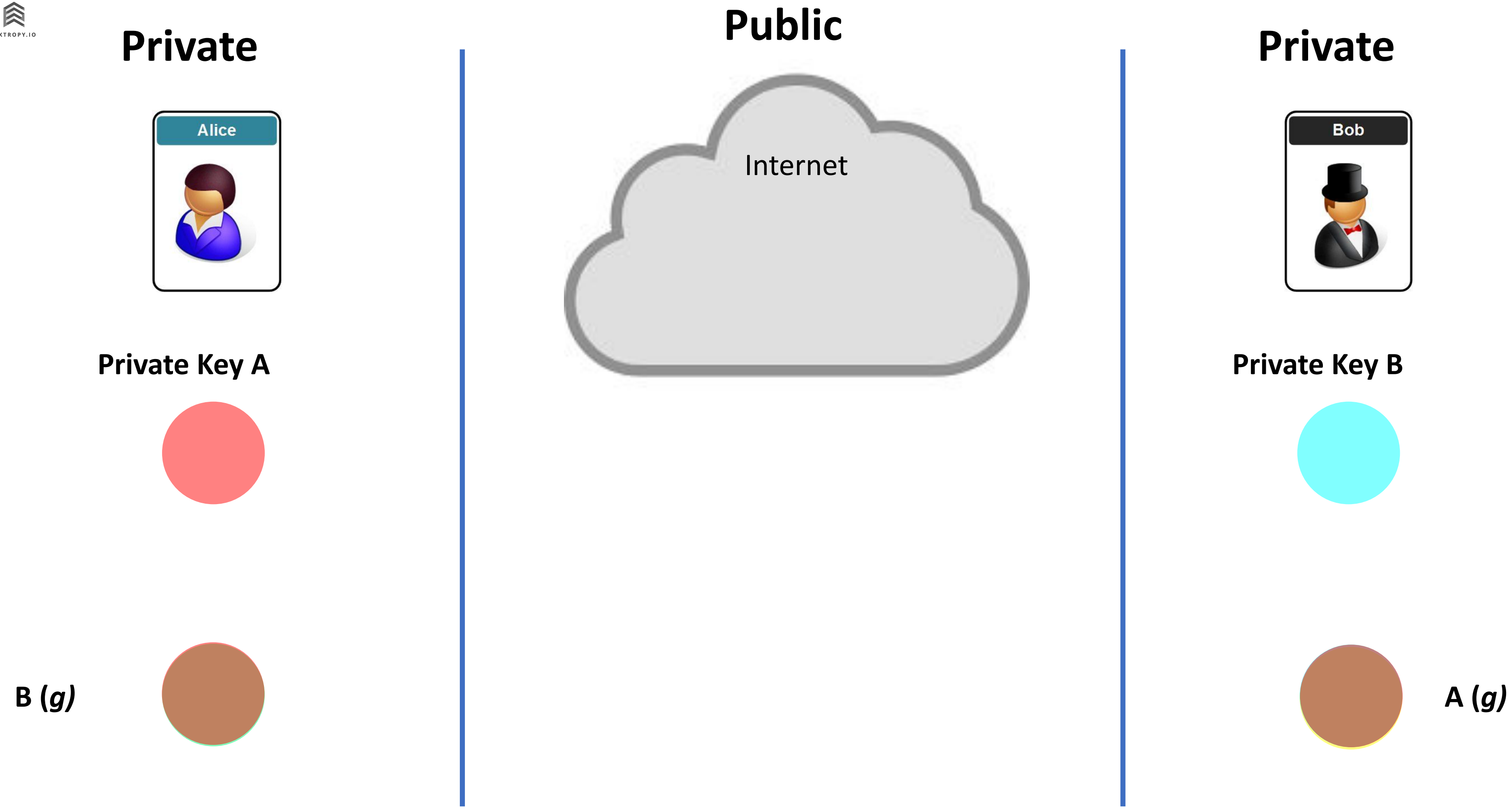
ey B

B (*g*)

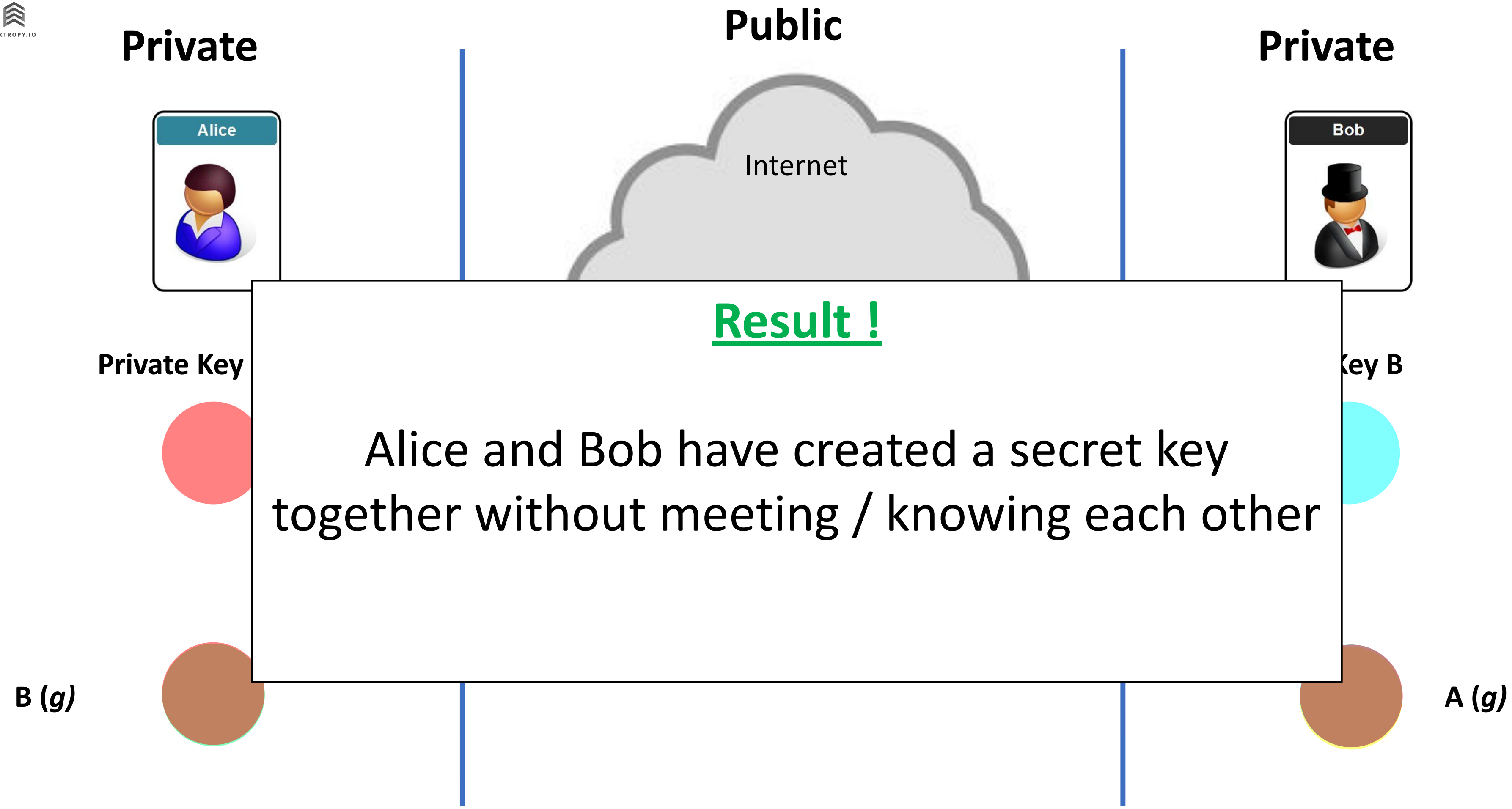
A (*g*)



Private
Public
Private
Alice
Internet
Bob
Private Key A
Private Key B
B (g)
A (g)



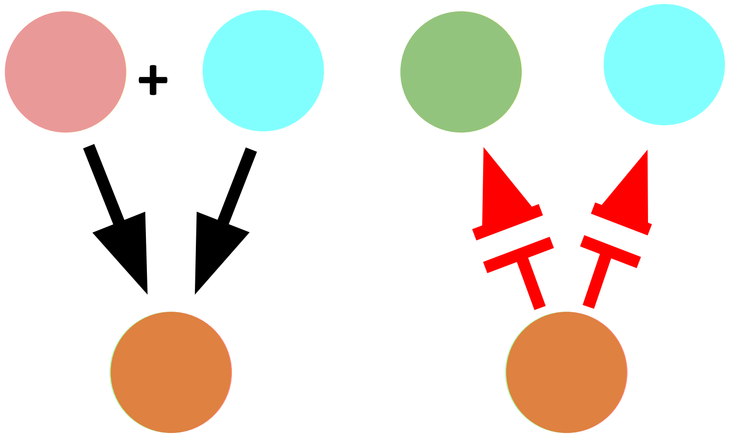
Private
Public
Private
Alice
Internet
Bob
Private Key A
Private Key B
B (g)
A (g)


Private
Public
Private
Alice
Internet
Bob
Result !
Alice and Bob have created a secret key together without meeting / knowing each other
Private Key
Key B
B (g)
A (g)

EXTROPY.IO

But what is even more interesting ?

- You can mix two colors to obtain one
- But **you can't unmix** to find out the colors you used !



Diffie and Hellman (1976, p1, para. 6)

Public key distribution systems offer a different approach to eliminating the need for a secure key distribution channel. In such a system, two users who wish to exchange a key communicate back and forth until they arrive at a key in common. A third party eavesdropping on this exchange must find it computationally infeasible to compute the key from the information overheard. A possible solu-

EXTROPY.IO

Menezes, Van Oorstone, Van Stone (1996) – *Handbook of Applied Cryptography*

12.47 Protocol Diffie-Hellman key agreement (basic version)

SUMMARY: A and B each send the other one message over an open channel.
RESULT: shared secret K known to both parties A and B.

1. *One-time setup*. An appropriate prime p and generator α of $\mathbb{Z}_p^*$ $(2 \leq \alpha \leq p-2)$ are selected and published.
2. *Protocol messages*.

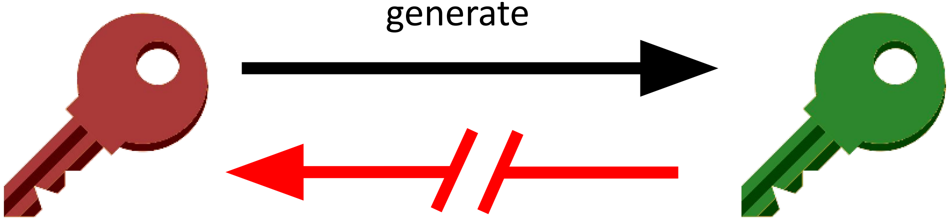
$$A \rightarrow B : \alpha^x \bmod p \quad (1)$$
$$A \leftarrow B : \alpha^y \bmod p \quad (2)$$

3. *Protocol actions*. Perform the following steps each time a shared key is required.
   (a) A chooses a random secret x, $1 \leq x \leq p-2$, and sends B message (1).
   (b) B chooses a random secret y, $1 \leq y \leq p-2$, and sends A message (2).
   (c) B receives α^x and computes the shared key as $K = (\alpha^x)^y \bmod p$.
   (d) A receives α^y and computes the shared key as $K = (\alpha^y)^x \bmod p$.

The Basics of Public Key Cryptography

Given a **Private Key**,you can derive a **Public Key**

But you can’t do the process in reverse (derive the Private Key from Public Key)



Private Key

Public Key

Keep it secret !

Available to everyone

Diffie and Hellman (1976, p1, para. 4)

without compromising the security of the system. In a *public key cryptosystem* enciphering and deciphering are governed by distinct keys, E and D, such that computing D from E is computationally infeasible (e.g., requiring 10^{100} instructions). The enciphering key E can thus be publicly disclosed without compromising the deciphering key D. Each user of the network can, therefore, place his enciphering key in a public directory. This enables any user of the system to send a message to any other user enciphered in such a way that only the intended receiver is able to decipher it. As such, a public key cryptosystem is a

1970's

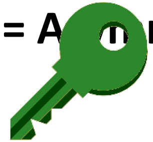
(1969)
The ARPANET (early Internet) was a p2p network

Diffie – Hellman Key Exchange

1976

Public Key Cryptography is invented !

(= Asymmetric Cryptography)



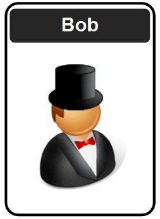
Public Key

Private Key

1970's

Secure Communication over Insecure Channel

<u>Public-Key Cryptography : 2 keys, one for encryption, one for decryption</u>



Private Key

<u>Step 1 :</u>

Alice and Bob

Create their own private / secret key

(privately, on their computer)

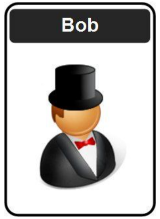
Private Key **A**

Malicious user

1970’s

Secure Communication over Insecure Channel

Public-Key Cryptography : 2 keys, one for encryption, one for decryption



Private Key

Public Key

Step 2 :

Alice and Bob

derive their public key based on their own private / secret key

(still privately, on their computer)

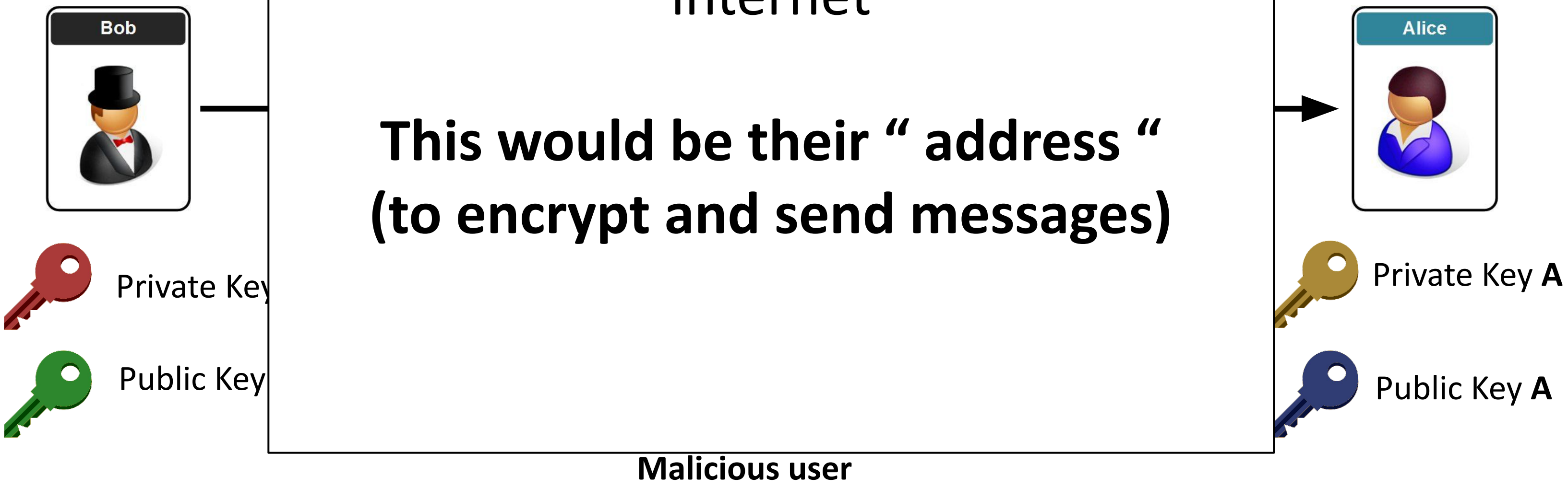
Private Key **A**

Public Key **A**

1970's

Secure Communication over Insecure Channel

Public-Key Cryptography : 2 keys, one for encryption, one for decryption



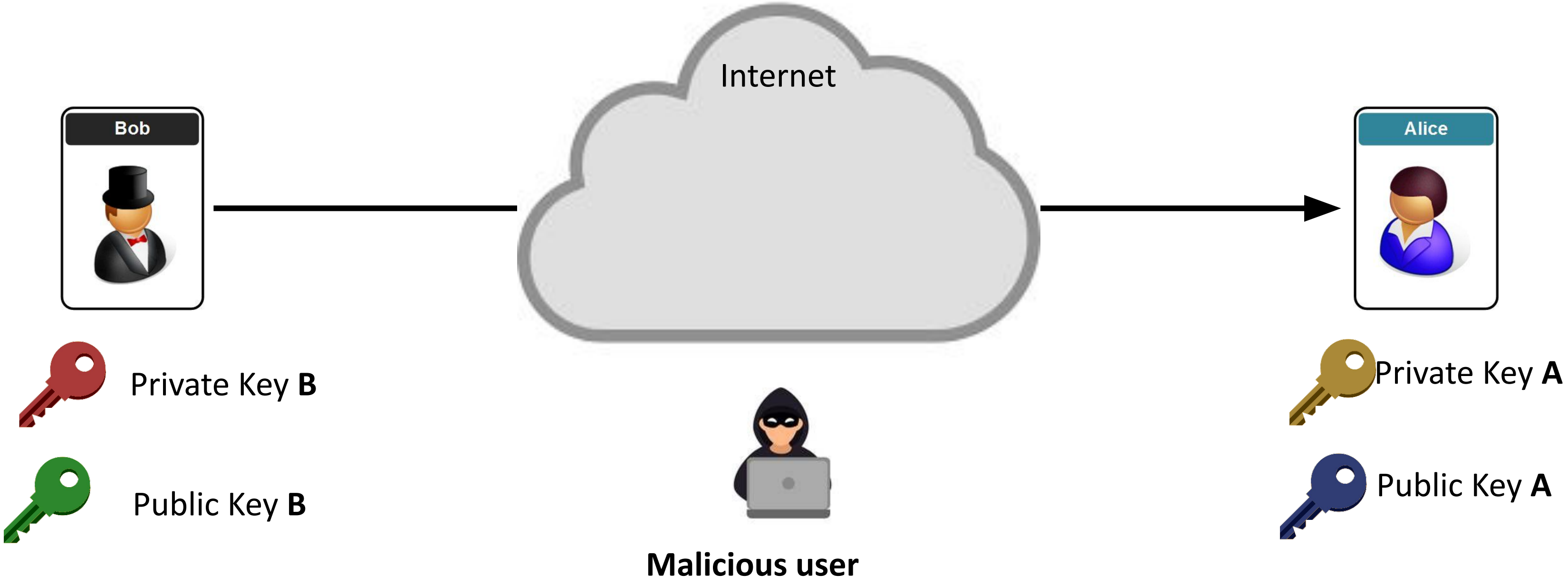
They publish their public key on the internet

This would be their " address "
(to encrypt and send messages)

1970's

Secure Communication over Insecure Channel

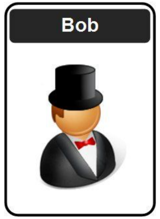
<u>Public-Key Cryptography :</u> 2 keys, one for encryption, one for decryption



1970's

Secure Communication over Insecure Channel

<u>Public-Key Cryptography :</u> 2 keys, one for encryption, one for decryption




Private Key

<u>Step 3 :</u>

Bob now can use Alice's Public Key to encrypt its message


Private Key **A**

<u>Message :</u>
Hello

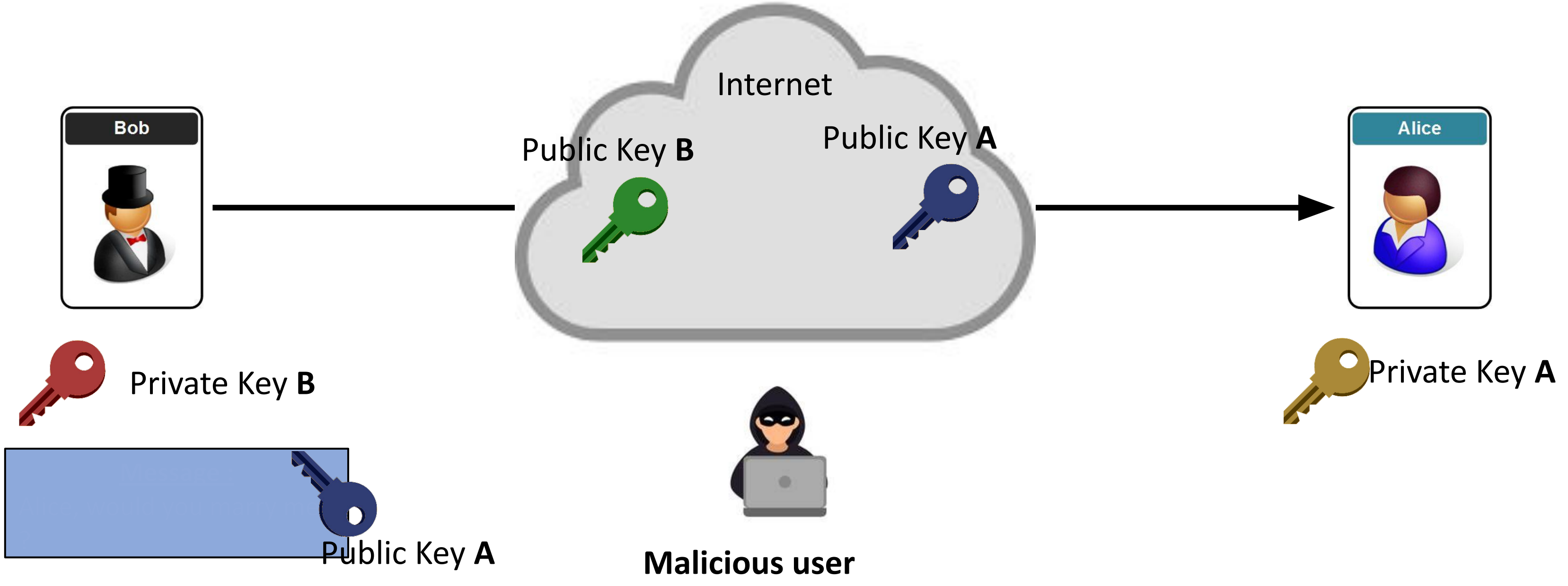
Let's Encrypt

Malicious user

1970's

Secure Communication over Insecure Channel

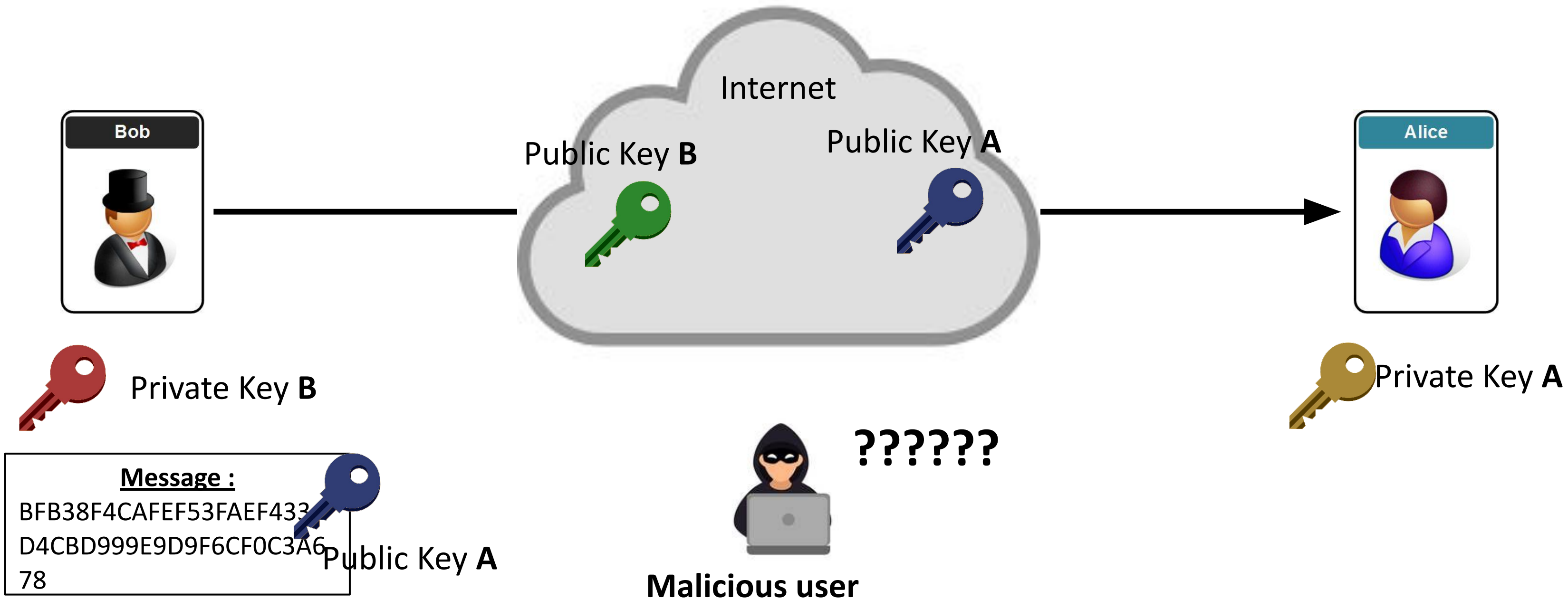
<u>Public-Key Cryptography :</u> 2 keys, one for encryption, one for decryption



1970's

Secure Communication over Insecure Channel

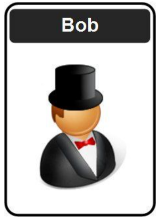
<u>Public-Key Cryptography :</u> 2 keys, one for encryption, one for decryption



1970's

Secure Communication over Insecure Channel

<u>Public-Key Cryptography :</u> 2 keys, one for encryption, one for decryption



Private Key

<u>Step 4 :</u>

Alice can now use her Private Key to decrypt Bob's message

Private Key **A**

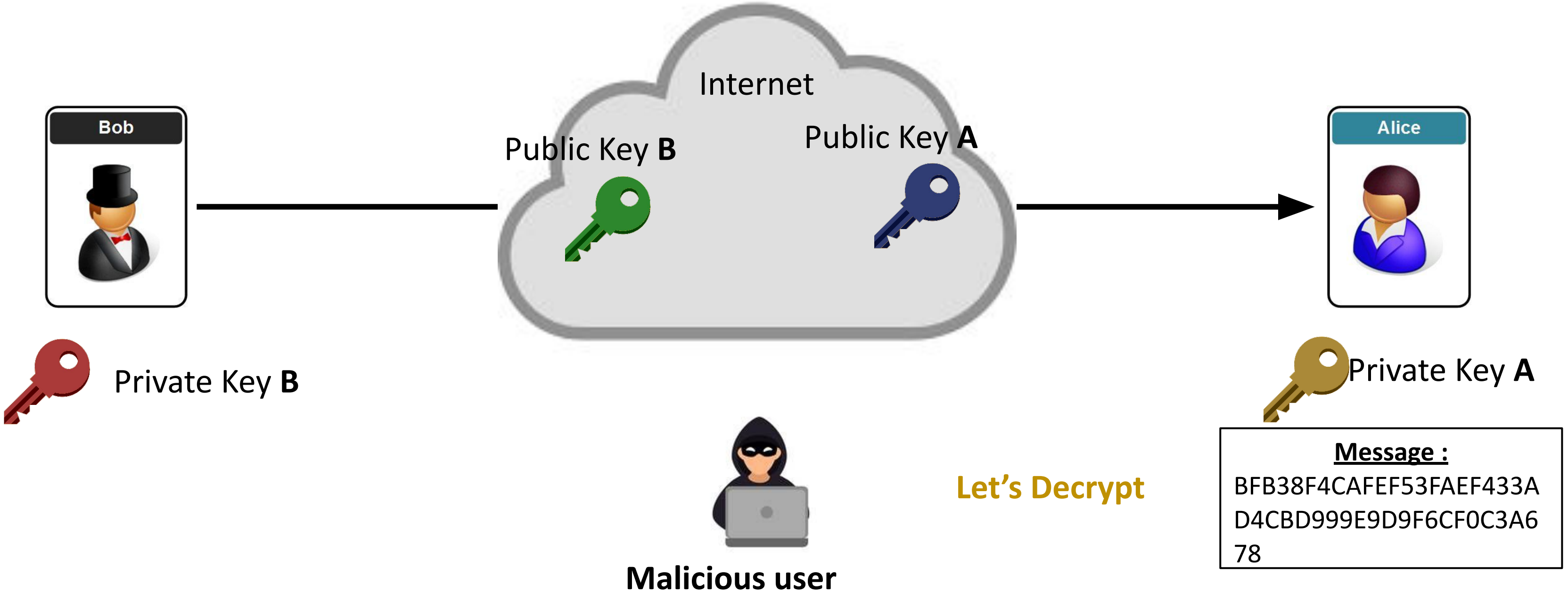
Malicious user

<u>Message :</u>
2987A1DB08C993478F7D9
BB00AE24CBF7DC2311BD0
E1

1970's

Secure Communication over Insecure Channel

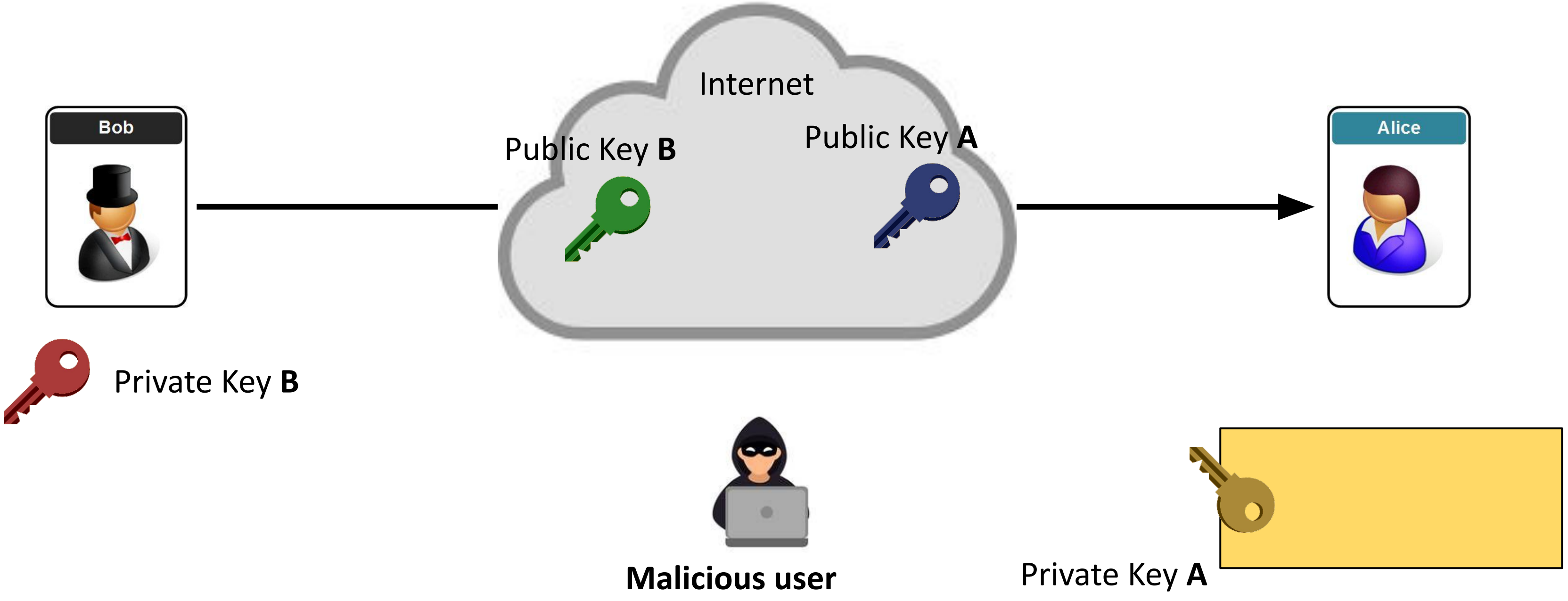
<u>Public-Key Cryptography :</u> 2 keys, one for encryption, one for decryption



1970's

Secure Communication over Insecure Channel

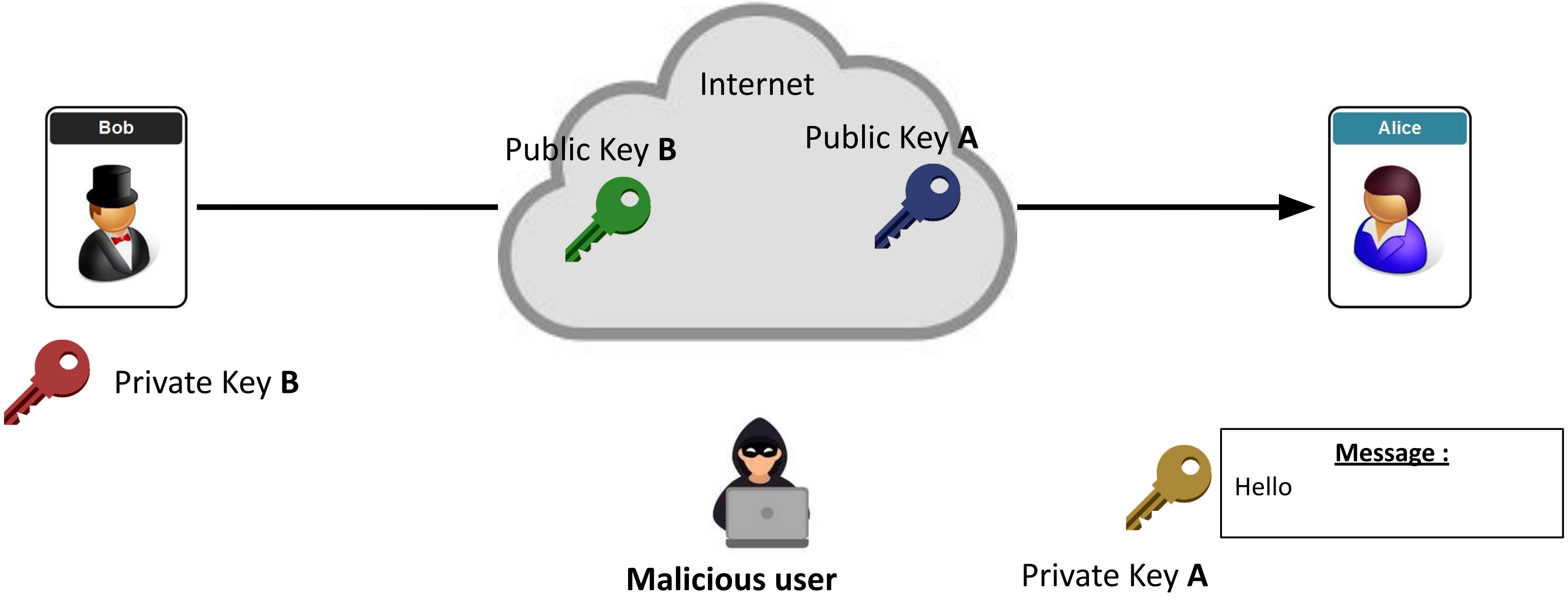
Public-Key Cryptography : **2 keys, one for encryption, one for decryption**



1970's

Secure Communication over Insecure Channel

Public-Key Cryptography : 2 keys, one for encryption, one for decryption



1970’s

Secure Communication over Insecure Channel

Asymm

We still have a problem here !

ey **A**

Ye

Private Key **B**

Malicious user

1970's

(1969)
The ARPANET
(early Internet) was a
p2p network

1976

Public Key Encryption solves :

<u>Problem 1 :</u> Key Management

- If I use a 3rd party to share my key, do I trust him ?

<u>Problem 2 :</u> Privacy

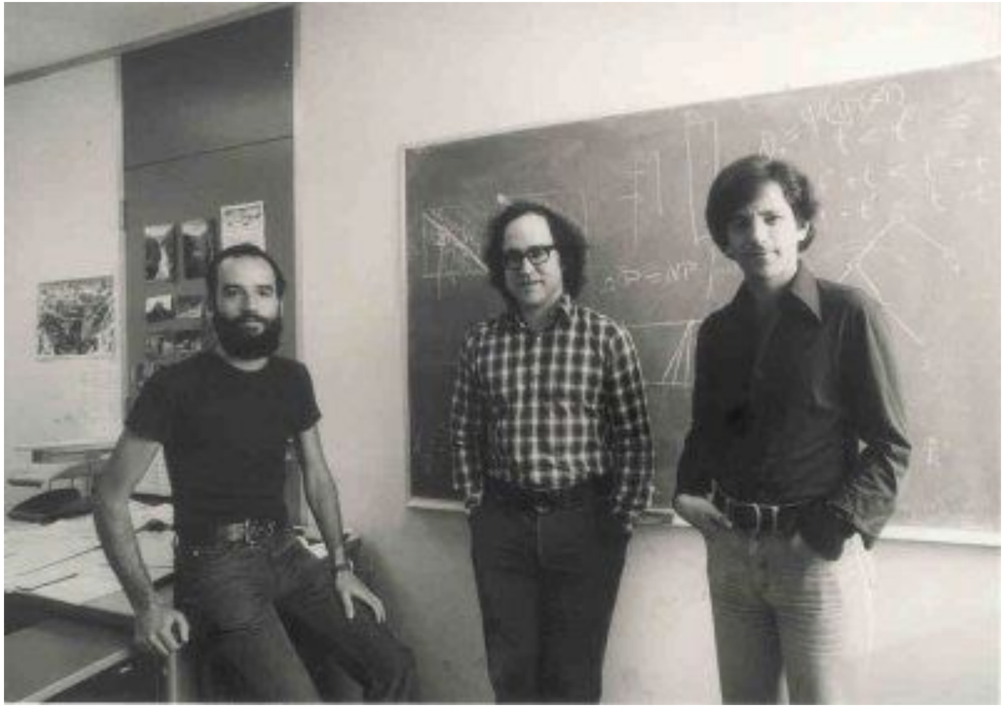
- How do I ensure that my message has not been modified ?

<u>Problem 3 :</u> Authenticity

- How do I ensure that the message comes from a legitimate person ?

1970's

Ron **R**ivest Adi **S**hamir Leonard **A**dleman → invent **RSA**



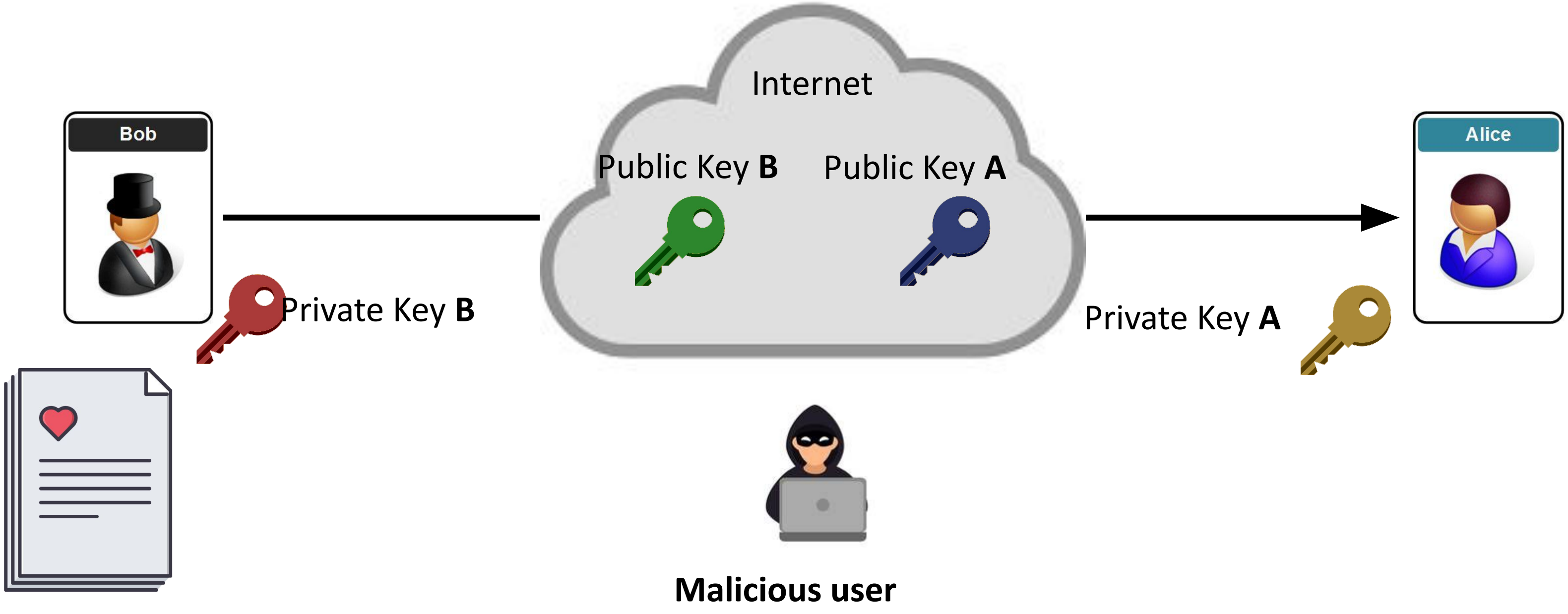
First implementation of

Digital Signature

1970's

Secure Communication over Insecure Channel

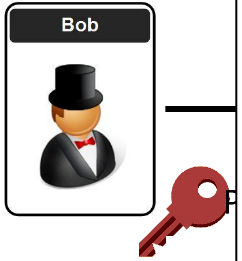
Digital Signature : Encrypt > Sign > Decrypt > Verify



1970's

Secure Communication over Insecure Channel

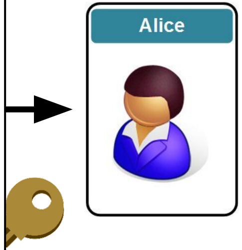
Digital Signature : Encrypt > Sign > Decrypt > Verify

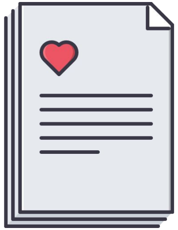


Step 1 :

Bob encrypts the document with his private key.

Thereby signing the document.



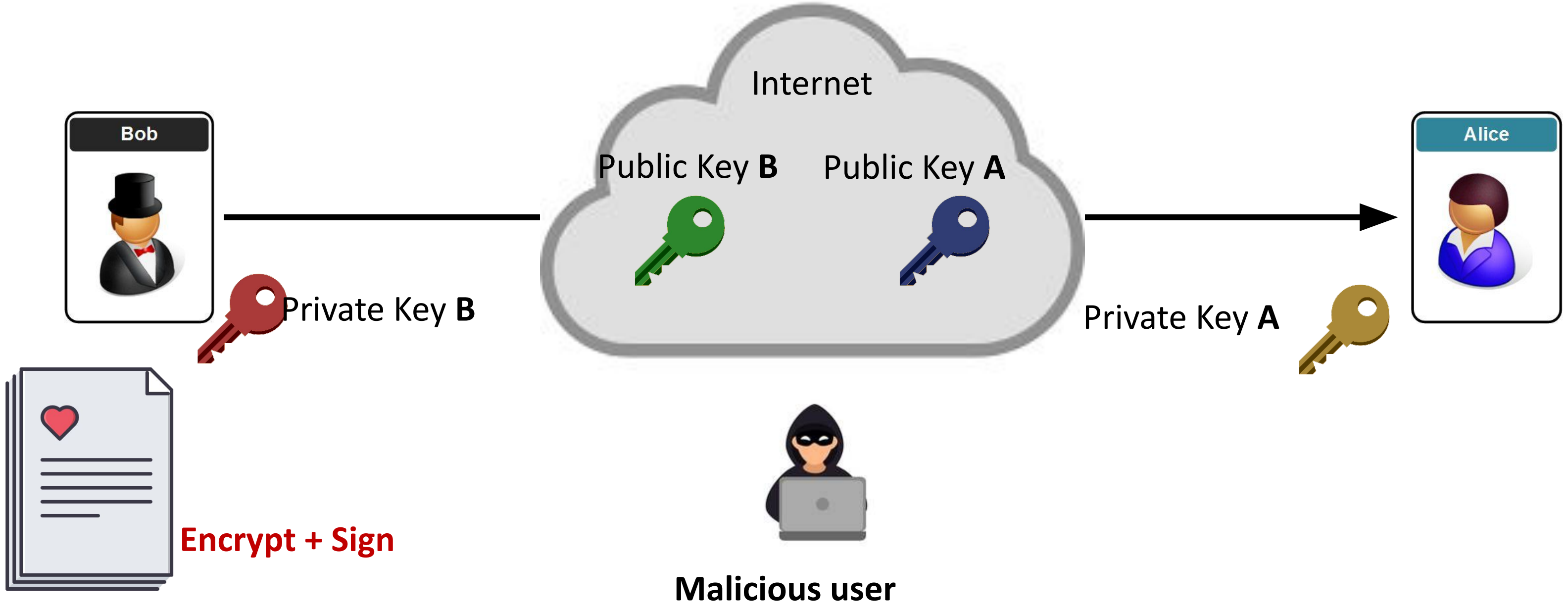


Malicious user

1970's

Secure Communication over Insecure Channel

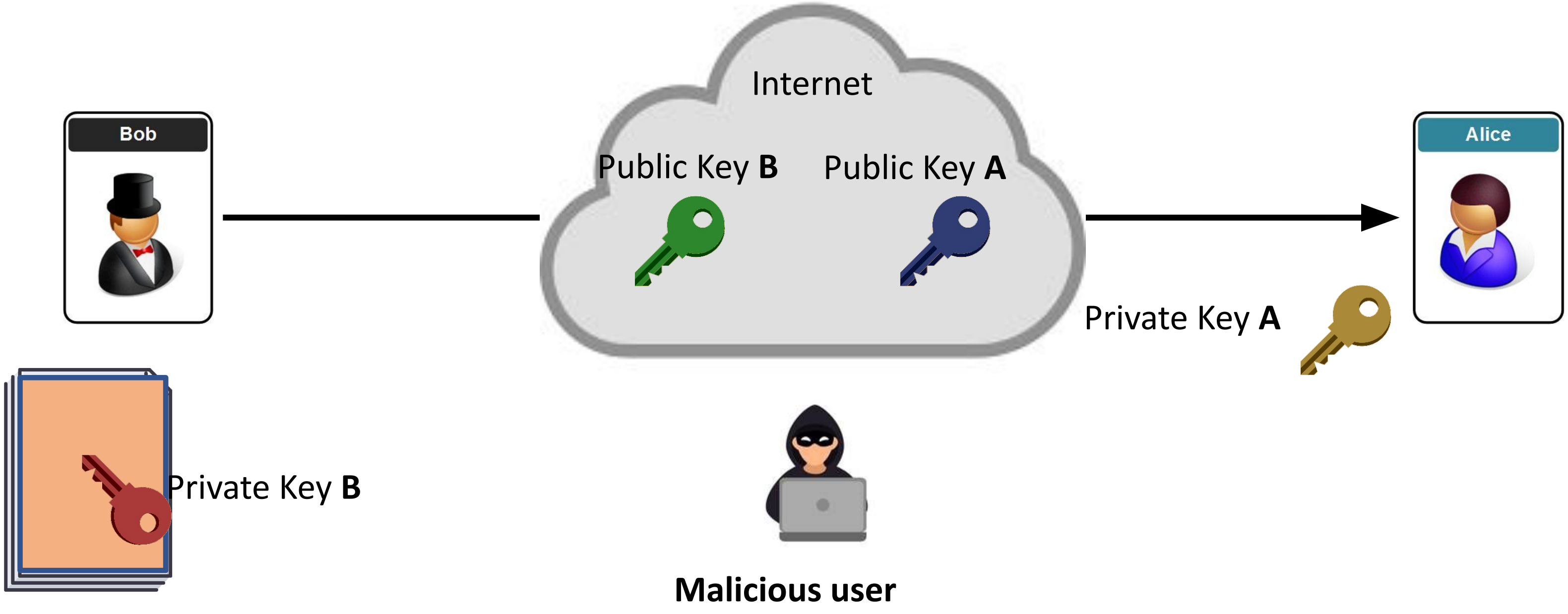
<u>Digital Signature :</u> Encrypt > Sign > Decrypt > Verify



1970's

Secure Communication over Insecure Channel

Digital Signature : Encrypt > Sign > Decrypt > Verify



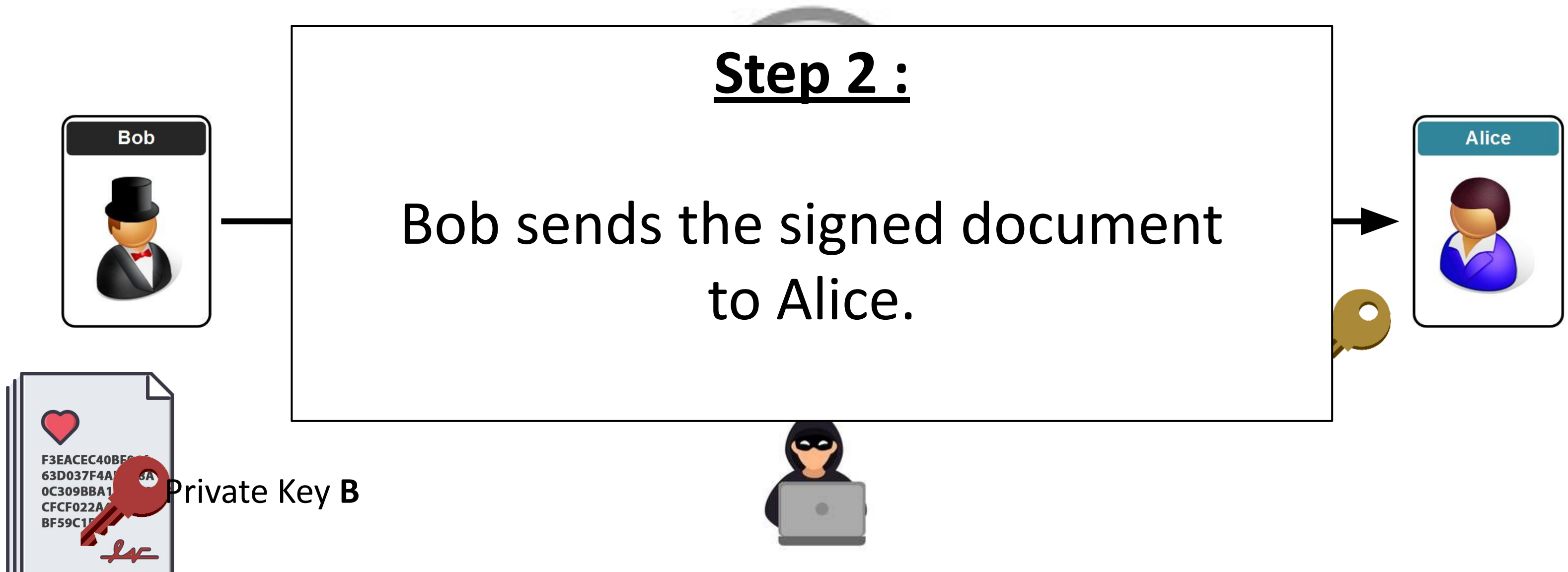
1970's

Secure Communication over Insecure Channel

<u>Digital Signature :</u> Encrypt > Sign > Decrypt > Verify

<u>Step 2 :</u>

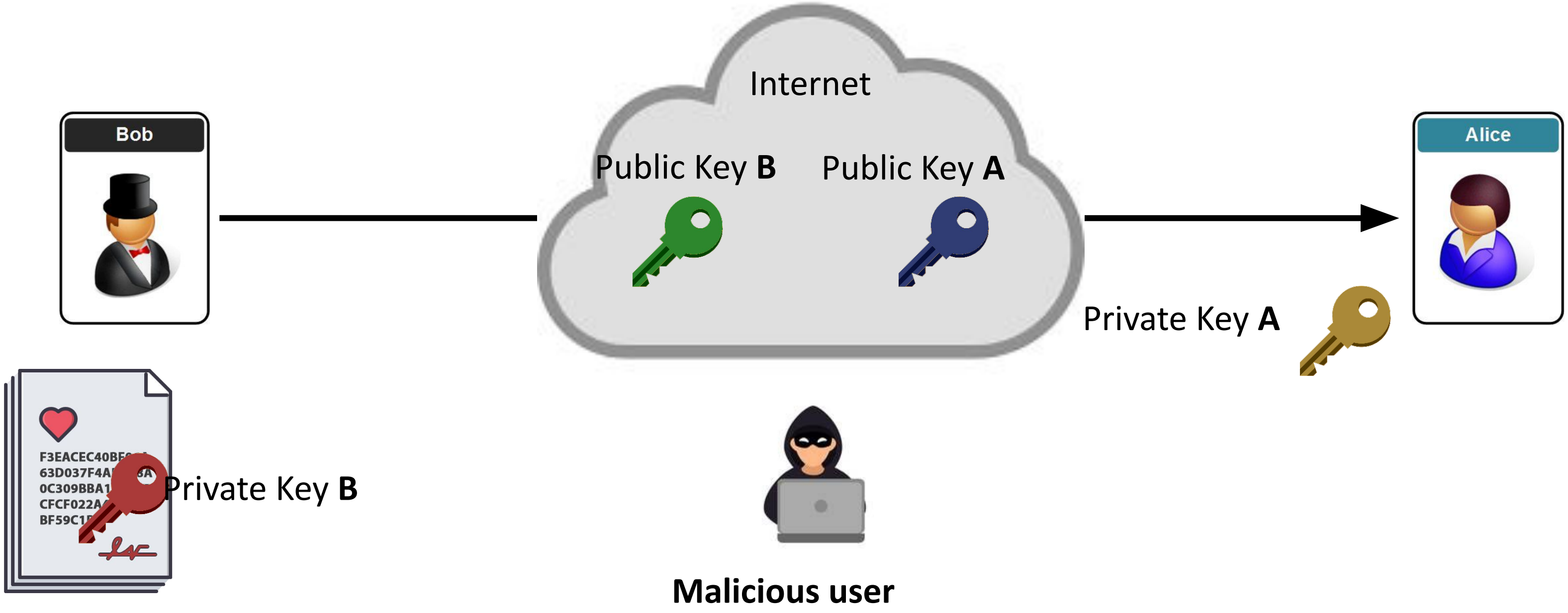
Bob sends the signed document to Alice.



1970's

Secure Communication over Insecure Channel

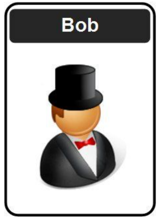
<u>Digital Signature :</u> Encrypt > Sign > Decrypt > Verify



1970's

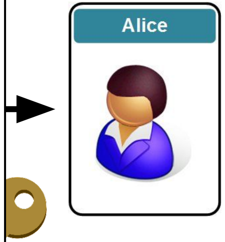
Secure Communication over Insecure Channel

<u>Digital Signature :</u> Encrypt > Sign > Decrypt > Verify



<u>Step 3 :</u>

Alice decrypts the document with Bob's public key, thereby verifying the signature.

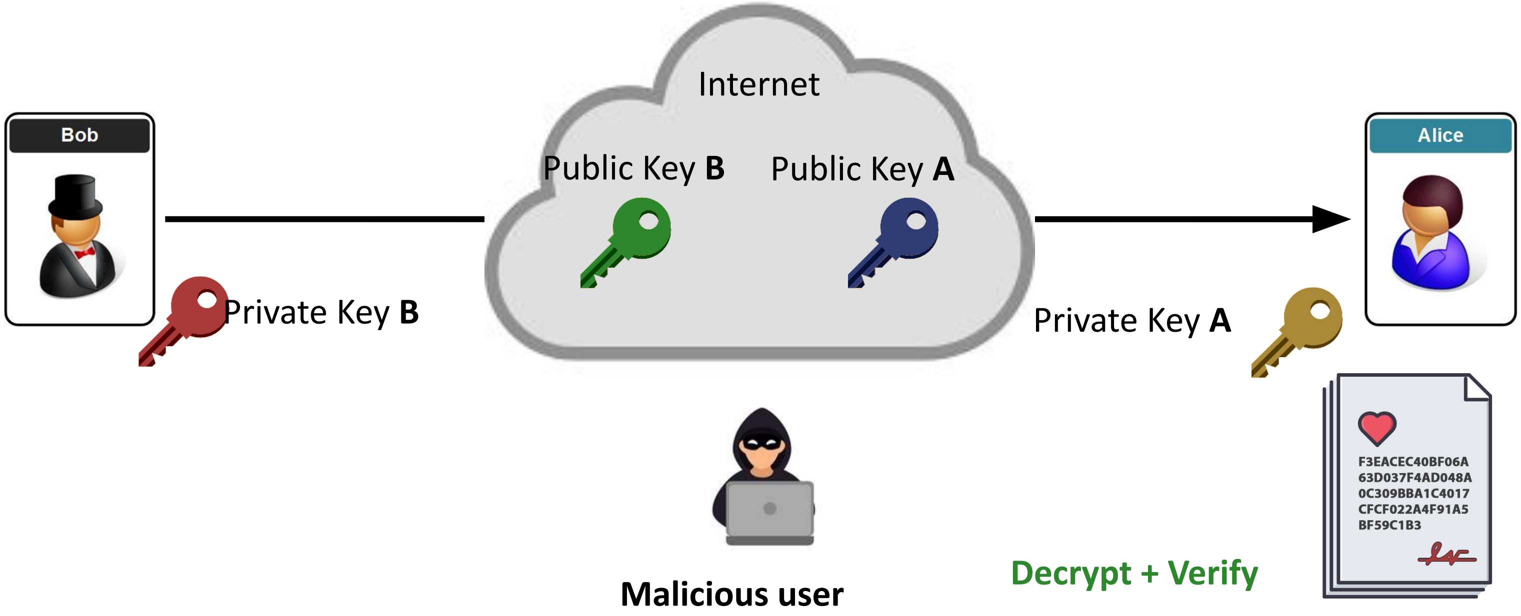


Malicious user

1970's

Secure Communication over Insecure Channel

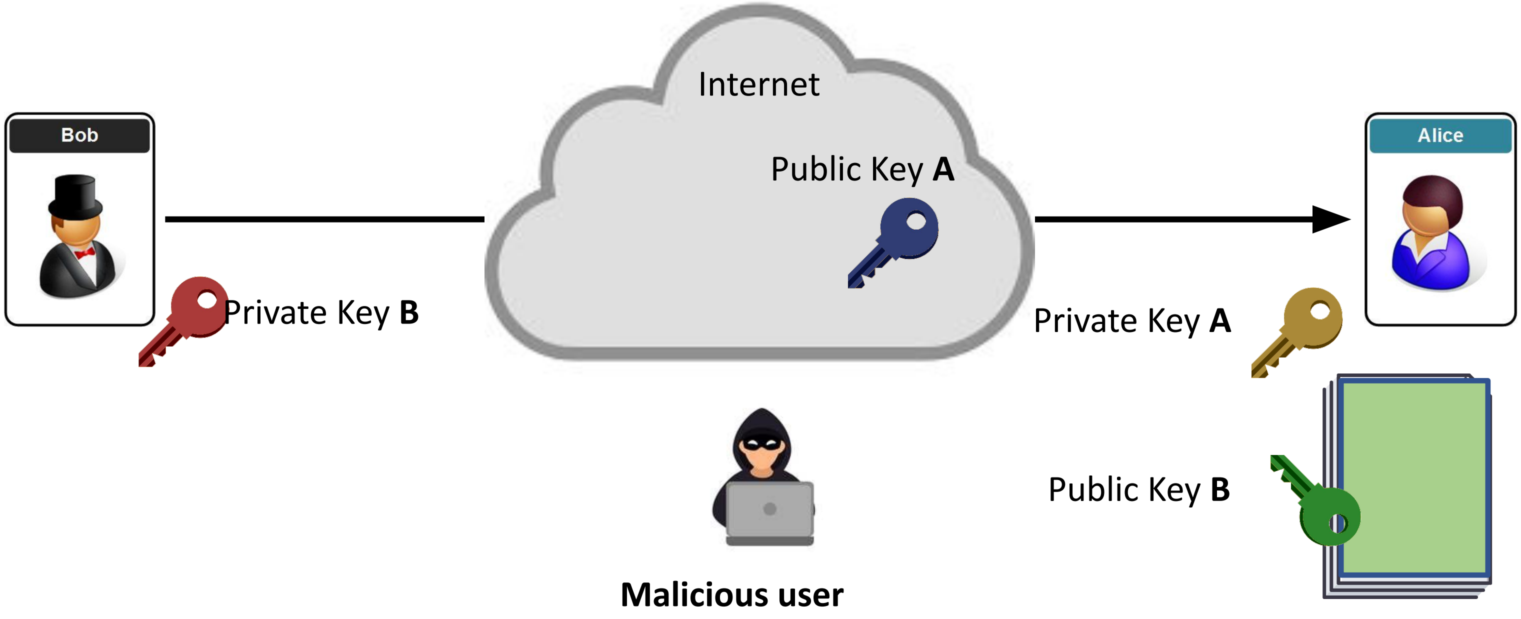
<u>Digital Signature :</u> Encrypt > Sign > Decrypt > Verify



1970's

Secure Communication over Insecure Channel

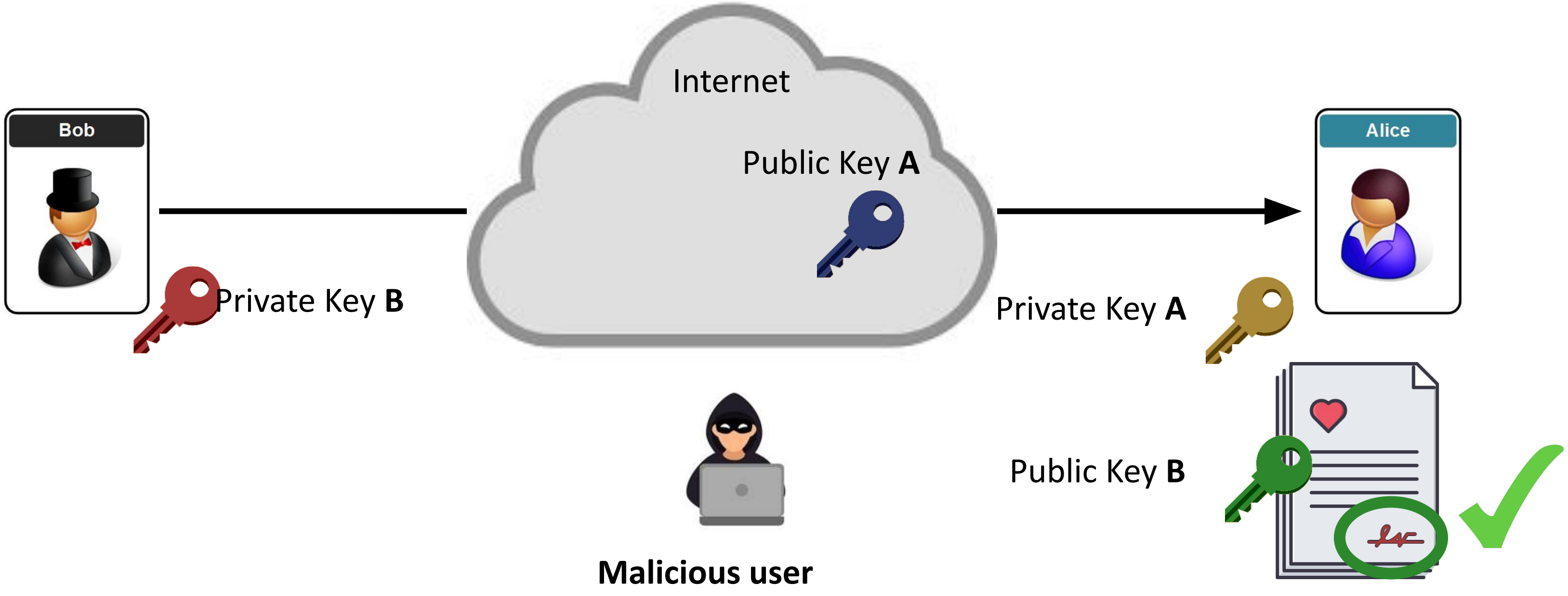
<u>Digital Signature :</u> Encrypt > Sign > Decrypt > Verify



1970's

Secure Communication over Insecure Channel

Digital Signature : Encrypt > Sign > Decrypt > Verify



EXTROPY.IO

Digital Signature : 4 properties

- **Authentic** : when Alice verifies the message with Bob's public key, she know that he signed the message.

- **Unforgeable** : only Bob knows his private key.

- **Not Reusable** : the signature is a <u>function of the document</u>. It can't be transferred to any other document.

- **Unalterable :** if there is any alteration to the document, the signature can no longer be verified with Bob's public key.

1970's

(1969)
The ARPANET
(early Internet) was a
p2p network

Public Key Cryptography
Diffie and Helman

1976

Digital Signature
Rivest, Shamir and Adelman

1979

Hash Functions
(Merkle Tree)
Ralph Merkle

EXTROPY.IO

1970's

Cryptography

TimeStamping

Ralph Merkle

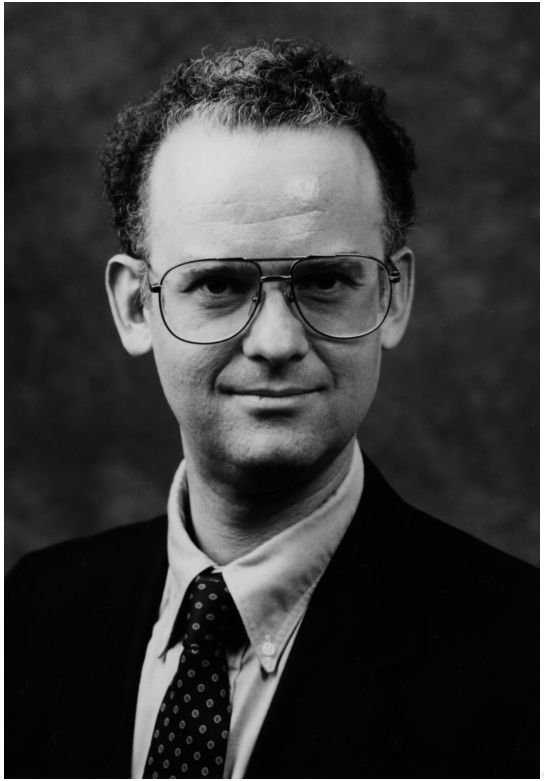
- Stanford University
- One of Larry Hellman's doctoral student.
- Invents :
  - Hashing Function
  - Merkle Tree

https://www.merkle.com

Cryptography

TimeStamping

1970’s



Objective : verify the presence of a document in a public directory

- Initially for digital certificates
- Undamaged and unaltered
- Securely and Efficiently

Cryptography

TimeStamping

0's

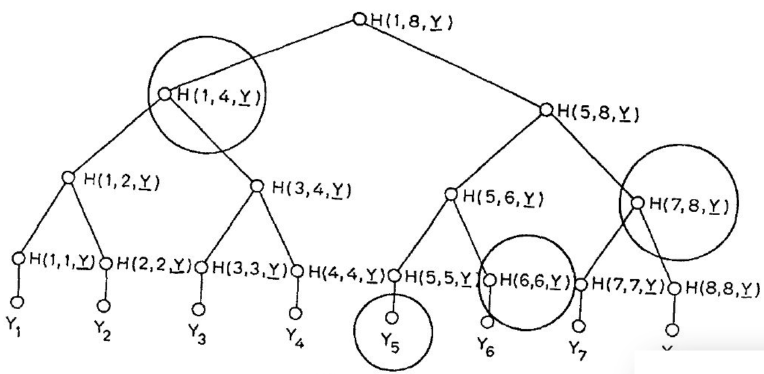
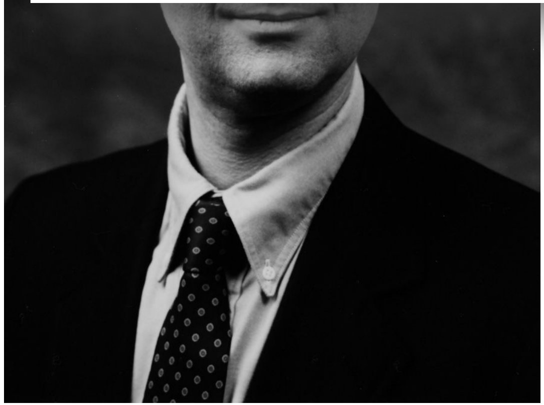


FIG. 1



graphy

Hash Functions (Merkle Tree)
Ralph Merkle

References

[1] W. Dai, "b-money," http://www.weidai.com/bmoney.txt, 1998.

[2] H. Massias, X.S. Avila, and J.-J. Quisquater, "Design of a secure timestamping service with minimal trust requirements," In *20th Symposium on Information Theory in the Benelux*, May 1999.

[3] S. Haber, W.S. Stornetta, "How to time-stamp a digital document," In *Journal of Cryptology*, vol 3, no 2, pages 99-111, 1991.

[4] D. Bayer, S. Haber, W.S. Stornetta, "Improving the efficiency and reliability of digital time-stamping," In *Sequences II: Methods in Communication, Security and Computer Science*, pages 329-334, 1993.

[5] S. Haber, W.S. Stornetta, "Secure names for bit-strings," In *Proceedings of the 4th ACM Conference on Computer and Communications Security*, pages 28-35, April 1997.

[6] A. Back, "Hashcash - a denial of service counter-measure," http://www.hashcash.org/papers/hashcash.pdf, 2002.

[7] R.C. Merkle, "Protocols for public key cryptosystems," In *Proc. 1980 Symposium on Security and Privacy*, IEEE Computer Society, pages 122-133, April 1980.

[8] W. Feller, "An introduction to probability theory and its applications," 1957.

EXTROPY.IO

Hash Function = Digital Fingerprint

Example :

(the example uses SHA256, try here > https://bit.ly/2YYMbF8

Input **Output**

Hash(**L**oughborough) → acb208d3ac02ab6d5a4…

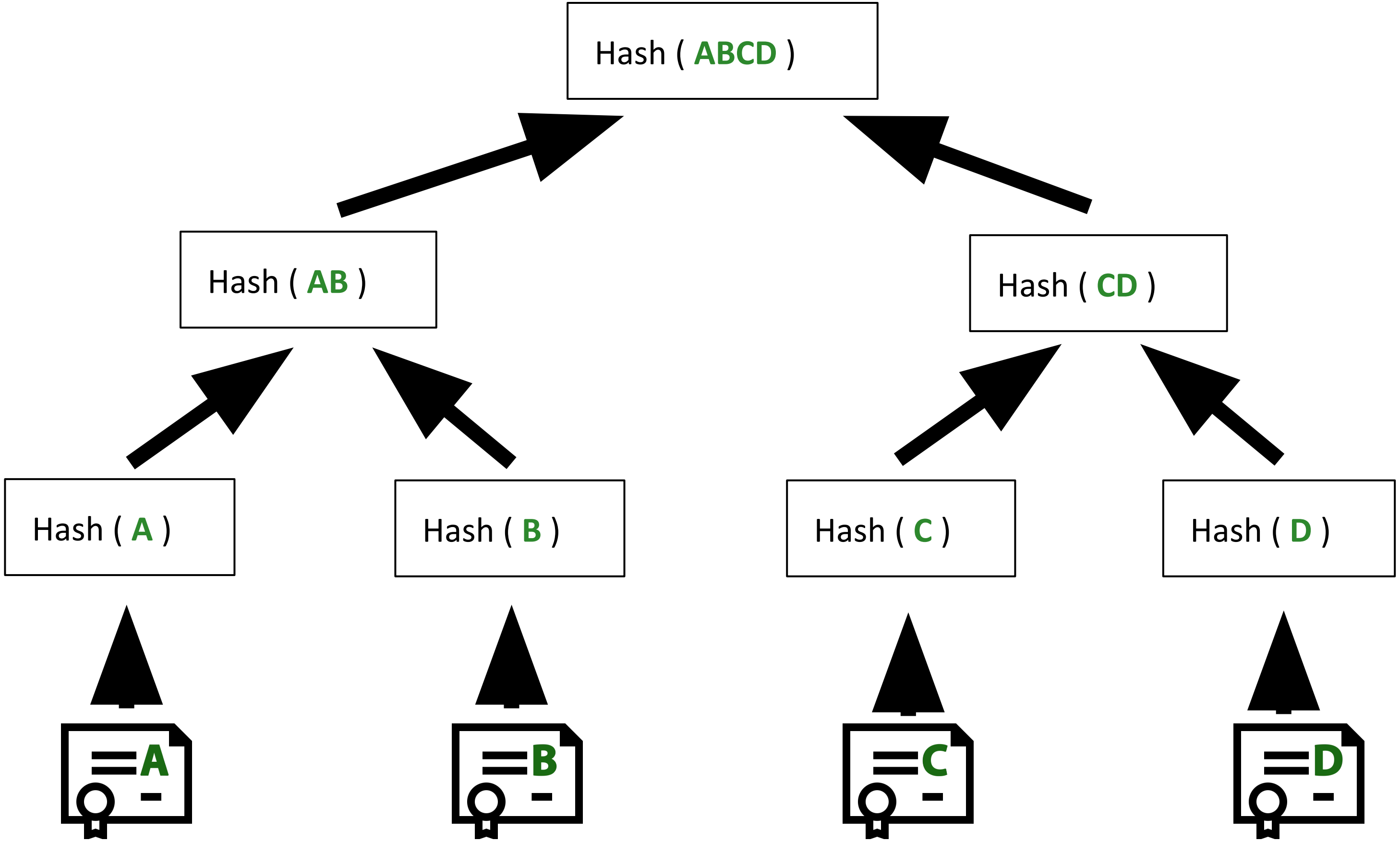
"L" **upper** case

Hash(**l**oughborough) → 1c2cc85d86f480bca5df0…

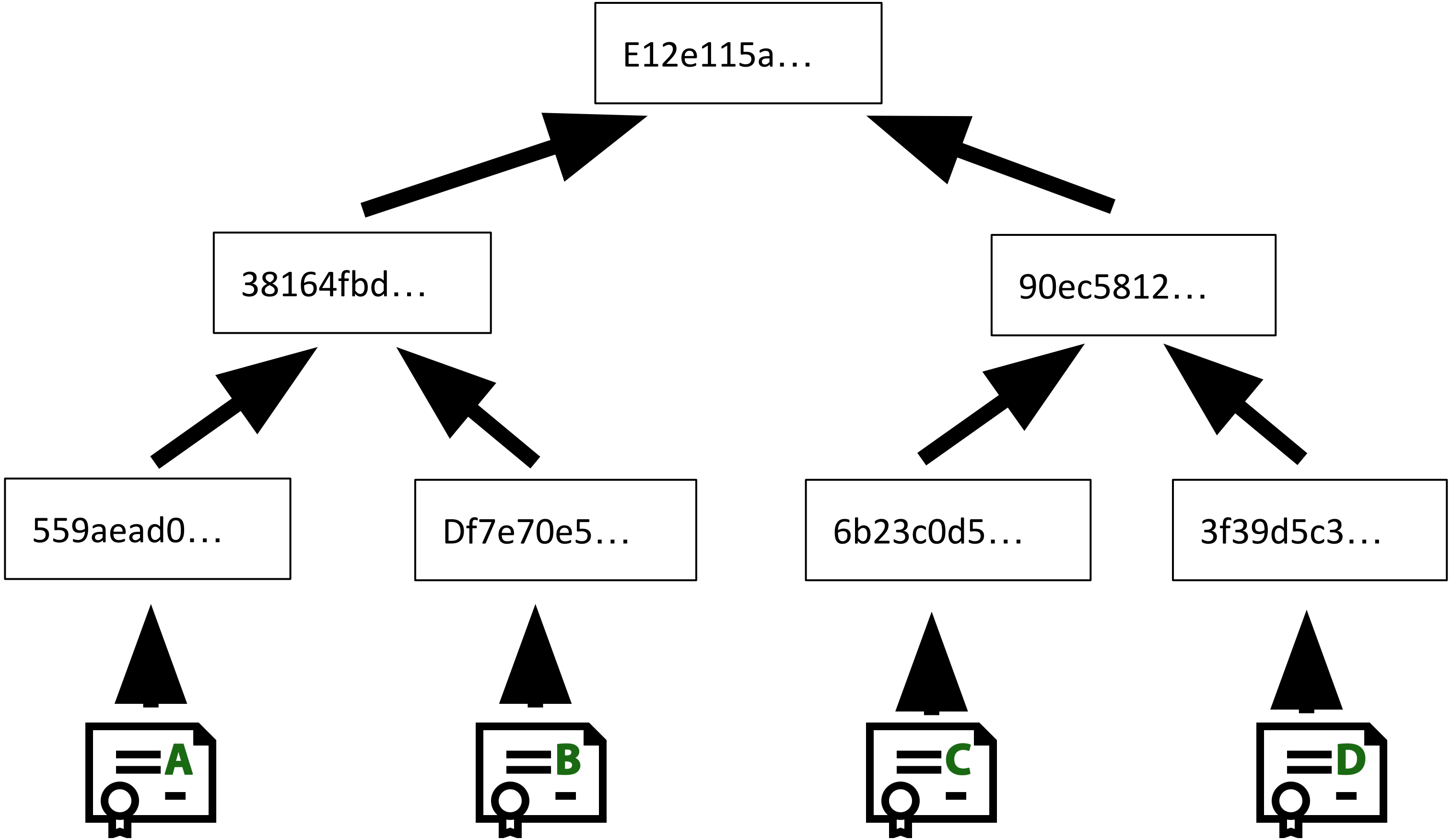
"l" **lower** case

A small change changes the whole digital fingerprint

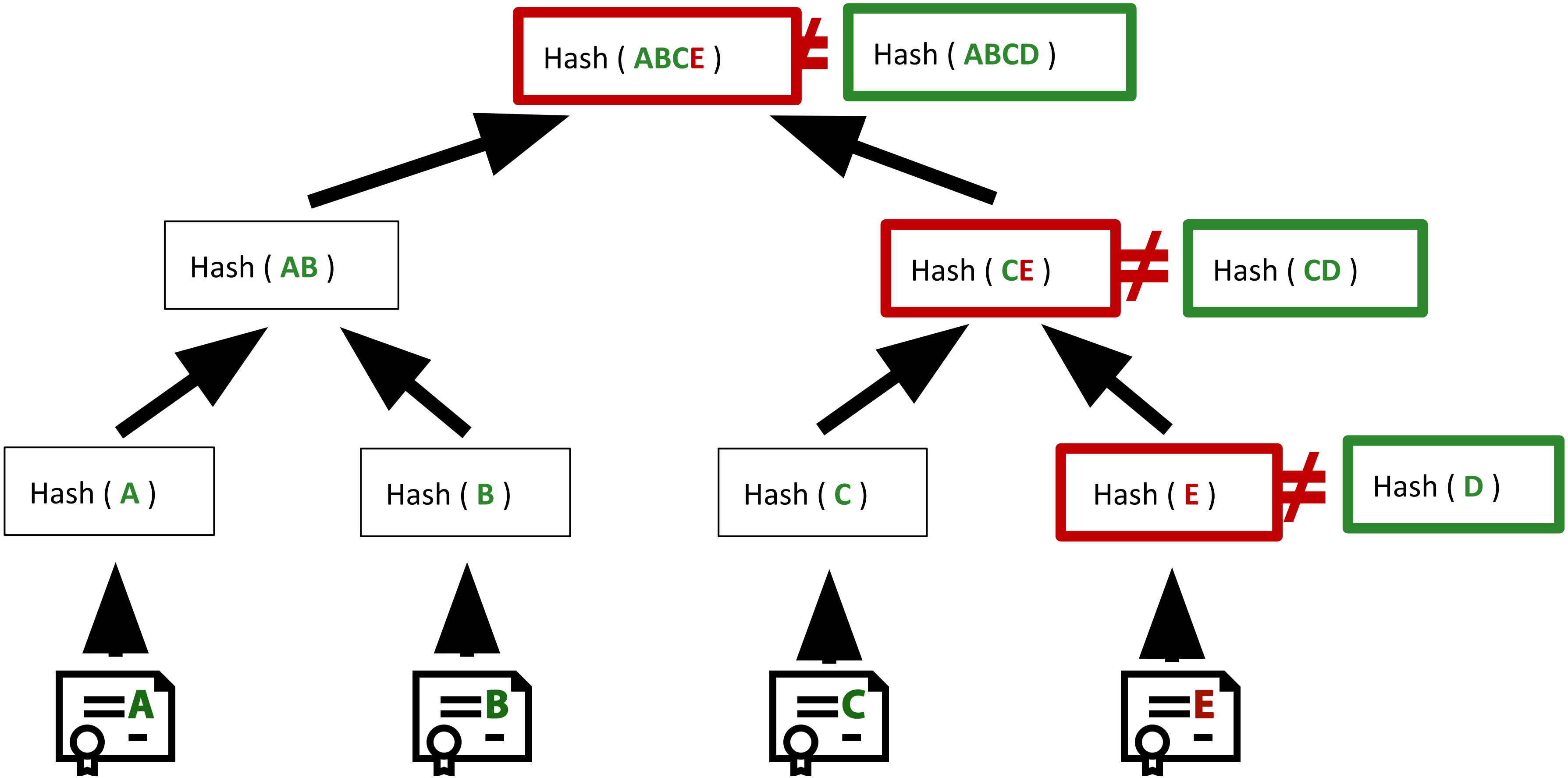
Merkle Tree (the basic)



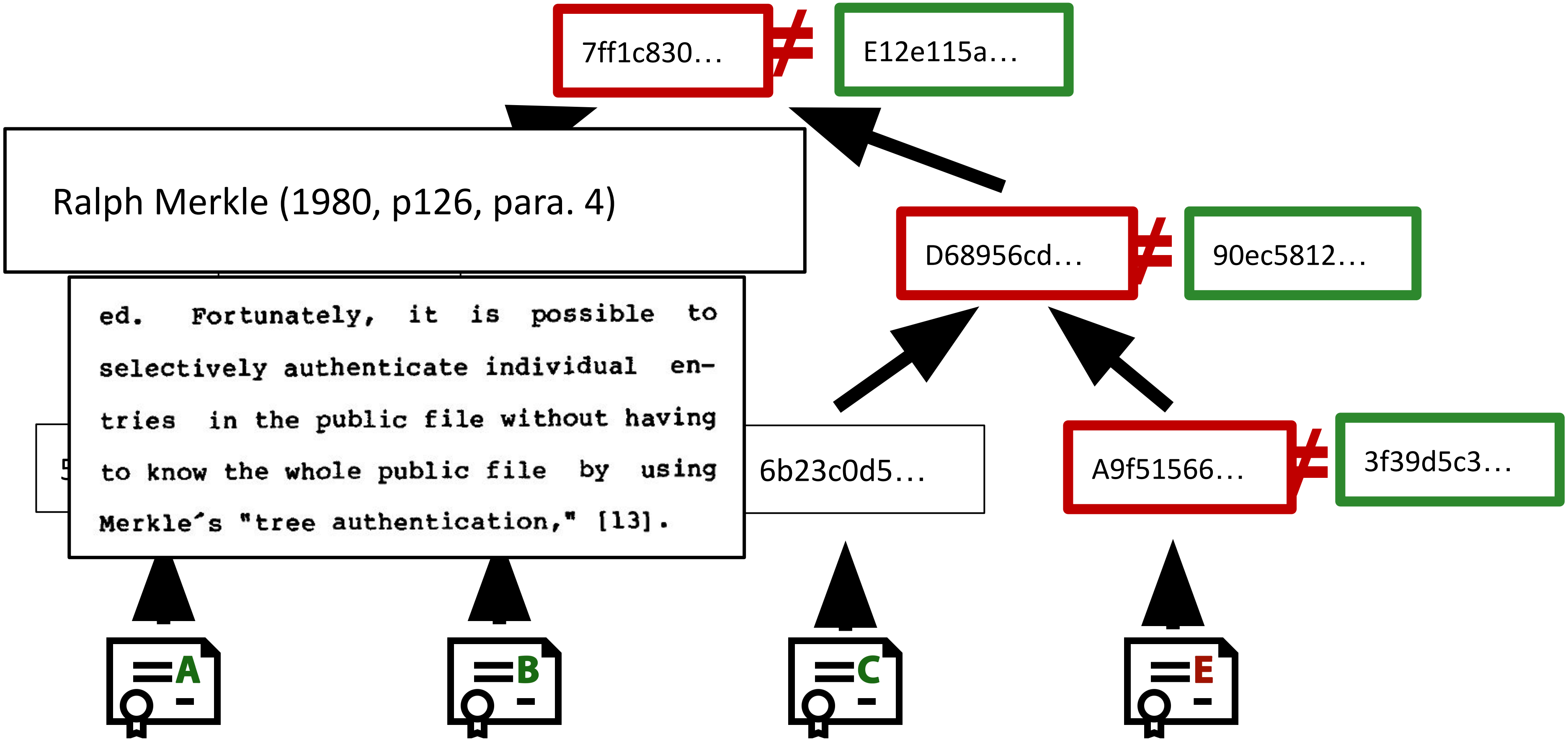
Merkle Tree (the basic)



Merkle Tree (the basic)



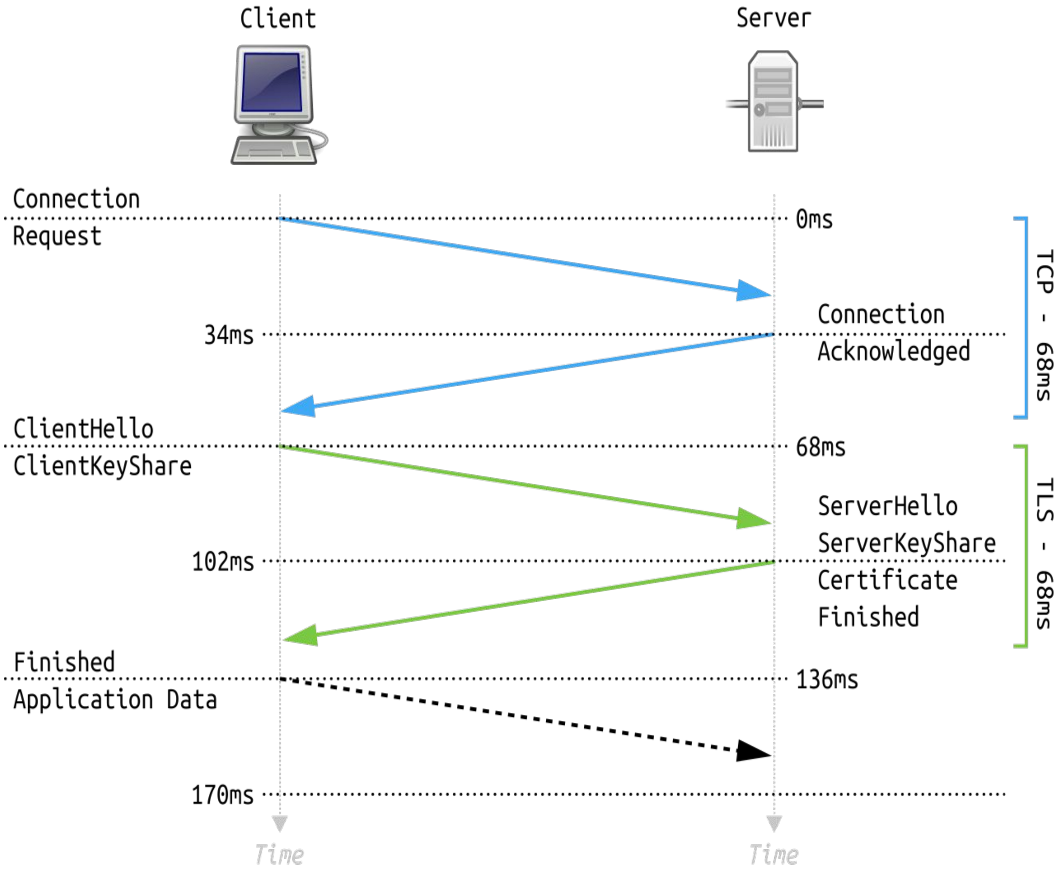
Merkle Tree (the basic)



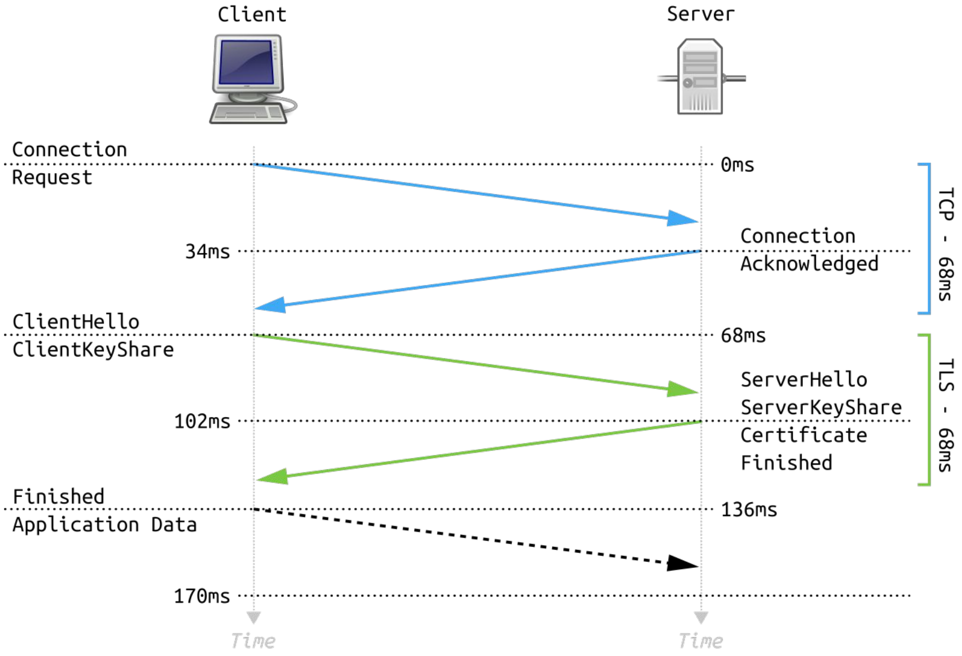
Ralph Merkle (1980, p126, para. 4)

ed. Fortunately, it is possible to selectively authenticate individual entries in the public file without having to know the whole public file by using Merkle's "tree authentication," [13].

Transport Layer Security



The server usually then provides identification in the form of a digital certificate. The certificate contains the server name, the trusted certificate authority (CA) that vouches for the authenticity of the certificate, and the server's public encryption key.
The client confirms the validity of the certificate before proceeding.



To generate the session keys used for the secure connection, the client either:

1. encrypts a random number (PreMasterSecret) with the server's public key and sends the result to the server (which only the server should be able to decrypt with its private key); both parties then use the random number to generate a unique session key for subsequent encryption and decryption of data during the session

2. uses Diffie–Hellman key exchange to securely generate a random and unique session key for encryption and decryption that has the additional property of forward secrecy: if the server's private key is disclosed in future, it cannot be used to decrypt the current session, even if the session is intercepted and recorded by a third party.

EXTROPY.IO

That is the cryptographic background, how did people try to use this technology ?

The development of

- Electronic cash
- Timestamping
- P2P Systems
- Consensus systems

EXTROPY.IO

Electronic cash

1980's

1983

Blind Signature for Untraceable Payments
David Chaum

E-Cash

BLIND SIGNATURES FOR UNTRACEABLE PAYMENTS

David Chaum

Department of Computer Science
University of California
Santa Barbara, CA

INTRODUCTION

Automation of the way we pay for goods and services is already underway, as can be seen by the variety and growth of electronic banking services available to consumers. The ultimate structure of the new electronic payments system may have a substantial impact on personal privacy as well as on the nature and extent of criminal use of payments. Ideally a new payments system should address both of these seemingly conflicting sets of concerns.

On the one hand, knowledge by a third party of the payee, amount, and time of payment for every transaction made by an individual can reveal a great deal about the individual's whereabouts, associations and lifestyle. For example, consider payments for such things as transportation, hotels, restaurants, movies, theater, lectures, food, pharmaceuticals, alcohol, books, periodicals, dues, religious and political contributions.

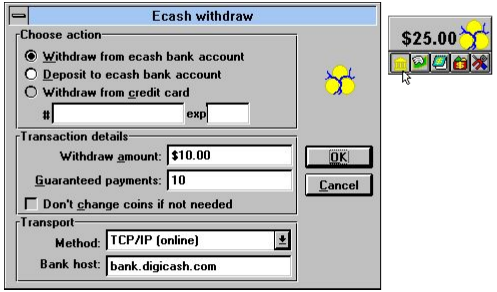
On the other hand, an anonymous payments systems like bank notes and coins suffers from lack of controls and security. For example, consider problems such as lack of proof of payment, theft of payments media, and black payments for bribes, tax evasion, and black markets.

A fundamentally new kind of cryptography is proposed here, which allows an automated payments system with the following properties:

Electronic cash

1980’s

E-Cash



Untraceable Electronic Cash
Chaum, Fiat and Naor

Extends to Offline payments

David Chaum founds the company

Digicash

1983

1988

1990

Blind Signature for Untraceable Payments
David Chaum

Starts with online transactions

1990’s

1991

Haber & Stornetta
How to TimeStamp a Digital Document

1992

Bayer, Haber & Stornetta
Improving the efficiency and reliability of digital time-stamping

Benaloh and De Mare
Efficient Broadcast TimeStamping

1997

Haber & Stornetta
Secure Name for BitStrings

1999

Massias, Avila and Quisquater
Design of a Secure Timestamping Service with minimal trust requirement

1990's

1991

Haber & Stornetta
How to TimeStamp a Digital Document

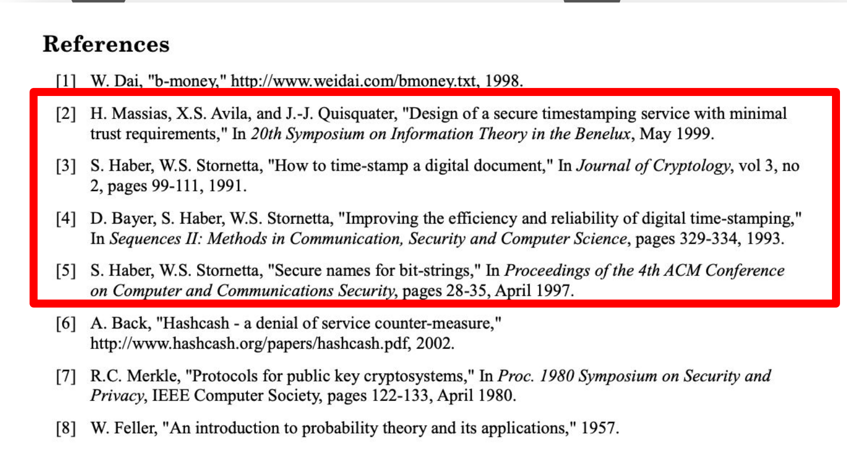
1992

Bayer, Haber & Stornetta
Improving the efficiency and reliability of digital time-stamping

Benaloh and De Mare
Efficient Broadcast TimeStamping

Haber & Stornetta
Secure Name for BitStrings

Massias, Avila and Quisquater
Design of a Secure Timestamping Service with minimal trust requirement



References

[1] W. Dai, "b-money," http://www.weidai.com/bmoney.txt, 1998.

[2] H. Massias, X.S. Avila, and J.-J. Quisquater, "Design of a secure timestamping service with minimal trust requirements," In *20th Symposium on Information Theory in the Benelux*, May 1999.

[3] S. Haber, W.S. Stornetta, "How to time-stamp a digital document," In *Journal of Cryptology*, vol 3, no 2, pages 99-111, 1991.

[4] D. Bayer, S. Haber, W.S. Stornetta, "Improving the efficiency and reliability of digital time-stamping," In *Sequences II: Methods in Communication, Security and Computer Science*, pages 329-334, 1993.

[5] S. Haber, W.S. Stornetta, "Secure names for bit-strings," In *Proceedings of the 4th ACM Conference on Computer and Communications Security*, pages 28-35, April 1997.

[6] A. Back, "Hashcash - a denial of service counter-measure," http://www.hashcash.org/papers/hashcash.pdf, 2002.

[7] R.C. Merkle, "Protocols for public key cryptosystems," In *Proc. 1980 Symposium on Security and Privacy*, IEEE Computer Society, pages 122-133, April 1980.

[8] W. Feller, "An introduction to probability theory and its applications," 1957.

1990's

TimeStamping

Consensus Algorithm

1991

Haber & Stornetta
How to TimeStamp a Digital Document

1992

Bayer, Haber & Stornetta
Improving the efficiency and reliability of digital time-stamping

Benaloh and De Mare
Efficient Broadcast TimeStamping

1993

Dwork and Naor
Pricing via Processing or Combatting Junk Mail

Pricing via Processing or Combatting Junk Mail*

Cynthia Dwork *   Moni Naor †

Abstract

We present a computational technique for combatting junk mail, in particular, and controlling access to a shared resource, in general. The main idea is to require a user to compute a moderately hard, but not intractable, function in order to gain access to the resource, thus preventing frivolous use. To this end we suggest several *pricing functions*, based on, respectively, extracting square roots modulo a prime, the Fiat-Shamir signature scheme, and the Ong-Schnorr-Shamir (cracked) signature scheme.

EXTROPY.IO

1990's

Electronic cash

TimeStamping

Consensus Algorithm

1991

Haber & Stornetta
How to TimeStamp a Digital Document

1992

Bayer, Haber & Stornetta
Improving the efficiency and reliability of digital time-stamping

Benaloh and De Mare
Efficient Broadcast TimeStamping

1993

Dwork and Naor
Pricing via Processing or Combatting Junk Mail

1996

How to Make a Mint: the Cryptography of Anonymous Electronic Cash - NSA

1997

HashCash
Adam Back

Haber & Stornetta
Secure Name for BitStrings

1998

B-Money
Wei Dai

Bit-Gold
Nick Szabo

1999

Auditable, Anonymous Electronic Cash
Sander & Ta-Shma

Massias, Avila and Quisquater
Design of a Secure Timestamping Service with minimal trust requirement

EXTROPY.IO

1990's

Electronic cash

TimeStamping

Consensus Algorithm

1991

Haber & Stornetta
How to TimeStamp a Digital Document

1992

Bayer, Haber & Stornetta
Improving the efficiency and reliability of digital time-stamping

Benaloh and De Mare
Efficient Broadcast TimeStamping

1993

Dwork and Naor
Pricing via Processing or Combatting Junk Mail

1996

How to Make a Mint: the Cryptography of Anonymous Electronic Cash - NSA

1997

HashCash
Adam Back

Haber & Stornetta
Secure Name for BitStrings

1998

B-Money
Wei Dai

Bit-Gold
Nick Szabo

1999

Auditable, Anonymous Electronic Cash
Sander & Ta-Shma

Massias, Avila and Quisquater
Design of a Secure Timestamping Service with minimal trust requirement

Napster

napster

1990's

Napster revolutionizes the music Industry

- Napster (1999) : peer-to-peer platform for sharing music
- Biggest Copyright Infringement
  - $26 million penalty fee to copyright owners
  - $10 million for future licensing royalties
- Shut Down on July 11, 2001

1990’s

Electronic cash

TimeStamping

Consensus Algorithm

1991

Haber & Stornetta
How to TimeStamp a Digital Document

1992

Bayer, Haber & Stornetta
Improving the efficiency and reliability of digital time-stamping

Benaloh and De Mare
Efficient Broadcast TimeStamping

1993

Dwork and Naor
Pricing via Processing or Combatting Junk Mail

1996

How to Make a Mint: the Cryptography of Anonymous Electronic Cash - NSA

1997

HashCash
Adam Back

Haber & Stornetta
Secure Name for BitStrings

1998

B-Money
Wei Dai

Bit-Gold
Nick Szabo

1999

Auditable, Anonymous Electronic Cash
Sander & Ta-Shma

Massias, Avila and Quisquater
Design of a Secure Timestamping Service with minimal trust requirement

Napster

21st Century

Peer-to-Peer networks emerge

SoulSeek

Freenet

GNutella

2000

2001

2004

BitTorrent

eDonkey

FastTrack / KaZaa

Early Attempts at Electronic Cash

“the one thing that’s missing is a reliable e-cash, whereby on the internet you can transfer funds from A to B without A knowing B or B knowing A” - Milton Friedman 1999

1998 - b-money - Wei Dai (http://www.weidai.com/bmoney.txt)

1998 - Bit Gold - Nick Szabo (https://nakamotoinstitute.org/bit-gold/)

Bitcoin QR Code

Satoshi Nakamoto is the name used by the presumed pseudonymous person or persons who developed bitcoin, authored the bitcoin white paper, and created and deployed bitcoin's original reference implementation

Early Bitcoin History

August 2008 - domain name bitcoin.org registered
October 2008 - A Peer-to-Peer Electronic Cash System posted to a cryptography mailing list
January 2009 - Software implementation released as open source

2010, the first known commercial transaction using bitcoin occurred when programmer Laszlo Hanyecz bought two Papa John's pizzas for 10,000 BTC

Differences between Bit Gold and Bitcoin

- PoW : Szabo’s concept of Bit Gold requires the amount of work required to create each piece to be quantifiable so that the market can price it.
- Szabo conceived Bit Gold as a reserve currency and not as a form of electronic money in itself.
- He thought his benchmark function impractical
- Double spending prevented by using a registry (distributed maintainers who vote)
- Nakamoto integrated the timechain and the registry
- Bitcoins are produced at a predictable rate and there is a finite supply

b-money

"A community is defined by the cooperation of its participants,
and efficient cooperation requires a medium of exchange (money) and a way to enforce contracts. Traditionally these services have been provided by the government or government sponsored institutions and only to legal entities. In this article I describe a protocol by which these services can be provided to and by untraceable entities." - Wei Dai 1998

b-money core concepts

- Requires a specified amount of computational work (Hashcash algorithm)
- The work done is verified by the community who update a collective ledger book.
- The worker is awarded funds for their effort.
- Exchange of funds is accomplished by collective book keeping and authenticated with cryptographic hashes.
- Contracts are enforced through the broadcast and signing of transactions with digital signatures (i.e., public key cryptography).

EXTROPY.IO

Some blockchain events since 2009

2014 - Ethereum created

2017 - ICO Boom / Alternatives to Ethereum

2018 - Crypto winter

2020 - DeFi summer

2021 - Rise of NFTs / Gaming

SUMMARY

Blockchains should be seen in the context of decentralisation

Cryptographic techniques have been crucial for solving the problems in implementing decentralised systems

Early electronic cash systems came part way to a practical implementation that could solve the double spend problem and achieve consensus across an open network.

Bitcoin solved these problems in 2009

Next lesson
Blockchain Theory 2