CS 4770: Cryptography

CS 6750: Cryptography and Communication Security

Alina Oprea
Associate Professor, CCIS
Northeastern University

April 2 2018

Schedule

- Grading project 2
  - Sign up on Doodle poll
  - Tue, Wed this week
- HW 4 (last written assignment)
  - It is out on Piazza
  - Due on Thu 04/12
- Programming project 3
  - Out on 04/12
  - Due on 04/26 (last day it can be accepted)
  - Grading on 04/27
- Final exam
  - 04/23, 1-3pm

Recap

- Digital signature schemes
  - Analogs of MACs in public-key setting
  - Public verifiability
  - Transferability
  - Non-repudiation
- Constructions
  - Hash-and-sign: Full-Domain Hash RSA
- PKI infrastructure
  - Distribute public keys
  - Hierarchical CA model
  - Single CA compromise can result in breaches
  - Revocation has a number of issues in practice

Secure communication on the Internet

- Generate public key, secret key pair (create digital identity)
  - Using Miller-Rabin primality testing
- Distribute the Public Key (announce identity)
  - Using digitally signed certificates and PKI
- Sender generates and sends secret key (initiate communication)
  - Using public-key CCA secure encryption
  - Or key exchange (Diffie-Hellman)
- Communicate securely (actual communication)
  - Using symmetric-key authenticated encryption

What Is SSL / TLS?

- Secure Sockets Layer and Transport Layer Security protocols
  - Same protocol design, different crypto algorithms
- De facto standard for Internet security
  - “The primary goal of the TLS protocol is to provide privacy and data integrity between two communicating applications”
- Deployed in every Web browser; also VoIP, payment systems, distributed systems, etc

SSL / TLS Guarantees

- End-to-end secure communications at transport layer in the presence of a network attacker
  - Attacker completely owns the network: controls Wi-Fi, DNS, routers, his own websites, can listen to any packet, modify packets in transit, inject his own packets into the network
- Properties
  - Authentication of server
  - Confidentiality of communication
  - Integrity against active attacks

History of the Protocol

- SSL 1.0 – internal Netscape design, early 1994
  - Lost in the mists of time
- SSL 2.0 – Netscape, Nov 1994
  - Several weaknesses
- SSL 3.0 – Netscape and Paul Kocher, Nov 1996
- TLS 1.0 – Internet standard, Jan 1999
  - Supersedes SSL: **SSL is known to be insecure**
  - Based on SSL 3.0, but not interoperable (uses different cryptographic algorithms)
- TLS 1.1 – Apr 2006
- TLS 1.2 – Aug 2008

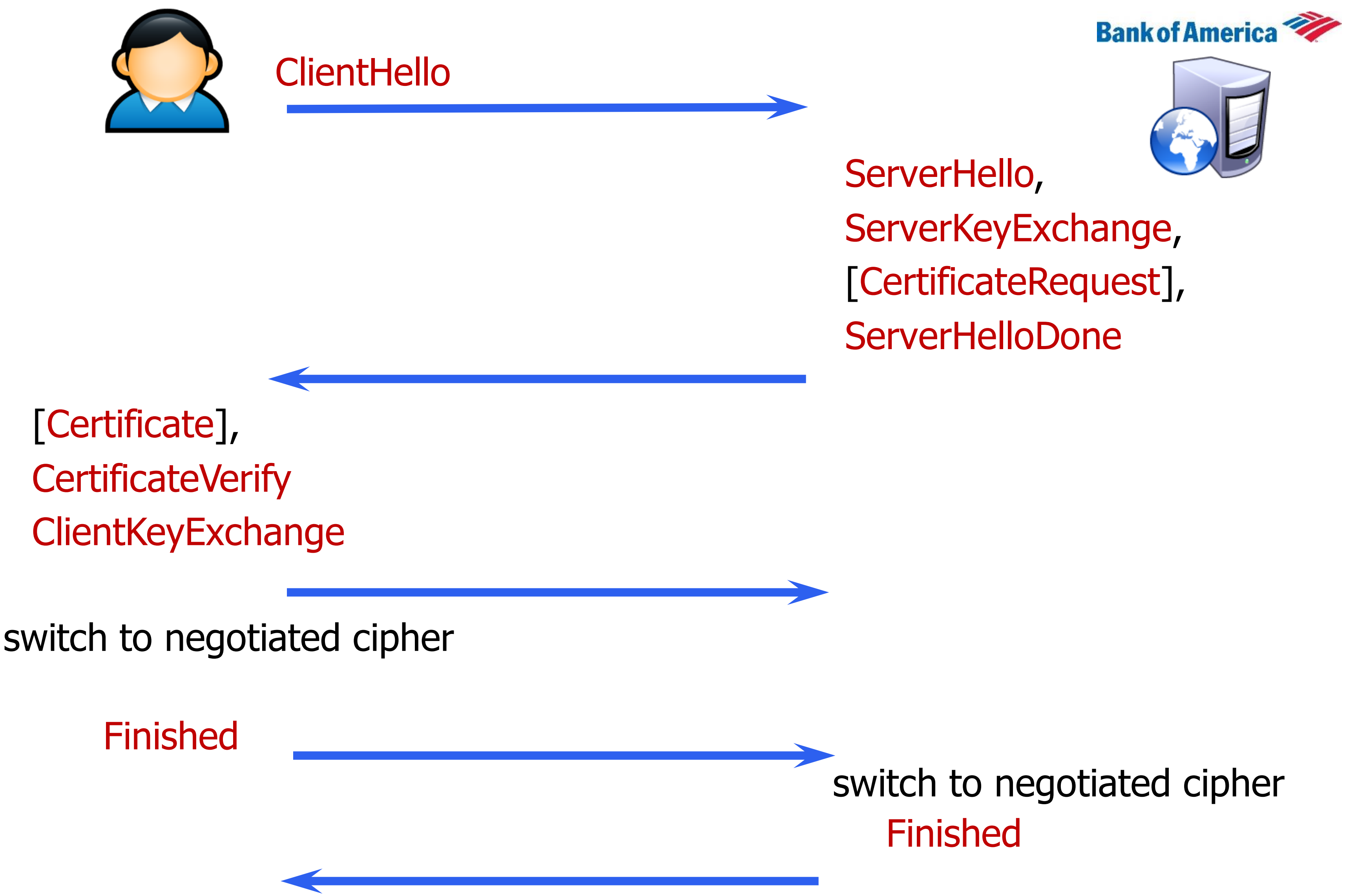
TLS Basics

- TLS consists of two protocols
- Handshake protocol
  - Session initiation by client
  - Uses public-key cryptography to establish several shared secret keys between the client and the server
  - Server must have an asymmetric keypair
    - X.509 certificates contain signed public keys rooted in PKI
- Record protocol
  - Uses the secret keys established in the handshake protocol to protect confidentiality and integrity of data exchange between the client and the server

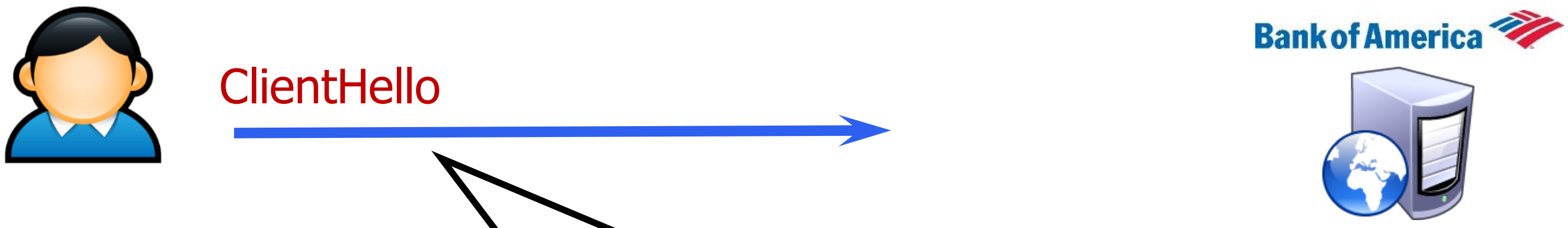
TLS Handshake Protocol

- Runs between a client and a server
  - Client = Web browser
  - Server = website
- Negotiate version of the protocol and the set of cryptographic algorithms to be used
  - Interoperability between different implementations
- Authenticate server
  - Use digital certificates to learn server’s public keys and verify the certificate
  - Client authentication is optional
- Use public keys to establish a shared secret

Handshake Protocol Structure



ClientHello



Client announces (in plaintext):

- Protocol version he is running
- Cryptographic algorithms he supports
- Fresh, random number

ClientHello (RFC)

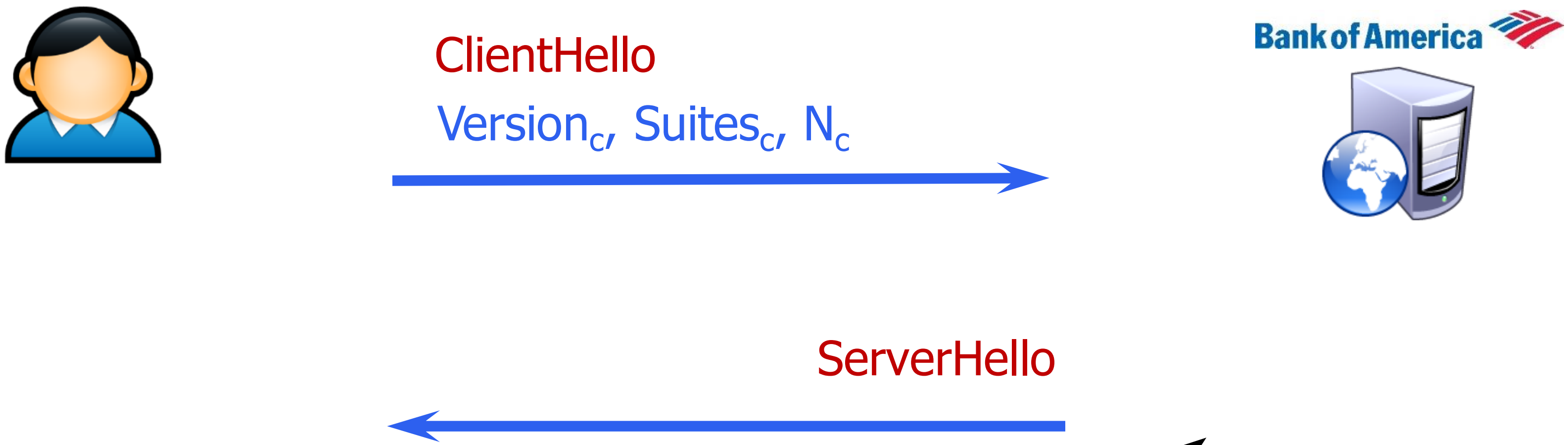
```
struct {
    ProtocolVersion client_version;
    Random random;
    SessionID session_id;
    CipherSuite cipher_suites;
    CompressionMethod compression_methods;
} ClientHello
```

Highest version of the protocol supported by the client

Session id (if the client wants to resume an old session)

Set of cryptographic algorithms supported by the client (e.g., RSA or Diffie-Hellman)

ServerHello



Server responds (<u>in plaintext</u>) with:

- Highest protocol version supported by both the client and the server
- Strongest cryptographic suite selected from those offered by the client
- Fresh, random number

ServerKeyExchange

ClientHello

Version$_C$, Suites$_C$, N$_C$

ServerHello

Version$_S$, Suites$_S$, N$_S$

ServerKeyExchange

Server sends his public-key certificate containing either his RSA, or his Diffie-Hellman public key (depending on chosen crypto suite)

ClientKeyExchange

Bank of America

ClientHello

Version_c, Suites_c, N_c

ServerHello

Version_S, Suites_S, N_S

ServerKeyExchange

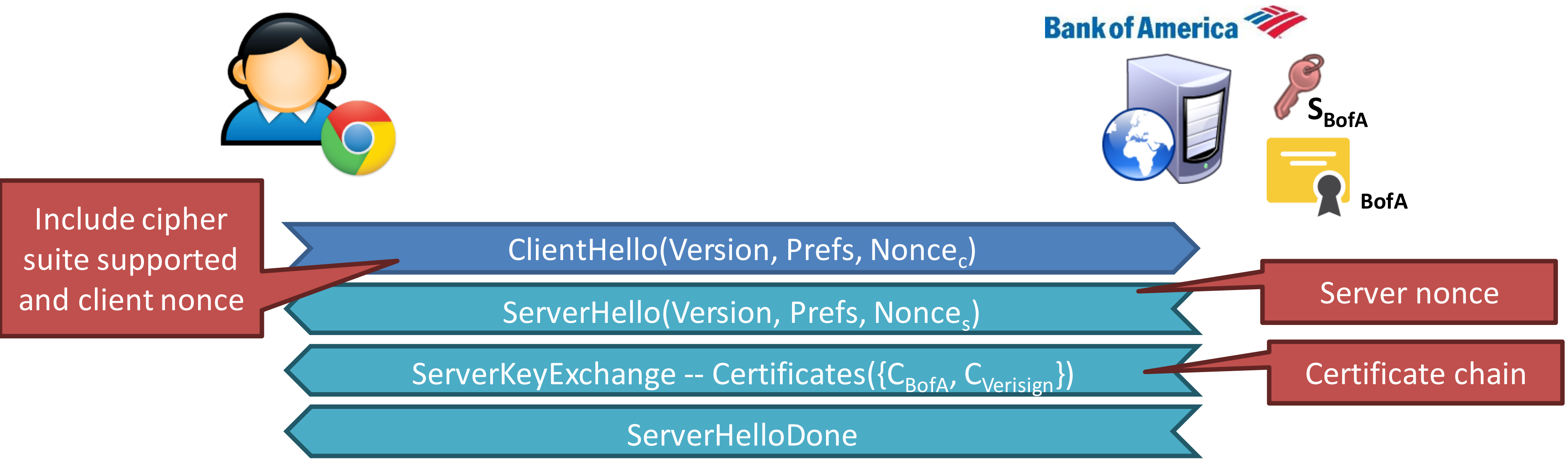
ServerHelloDone

CertificateVerify

ClientKeyExchange

The client generates secret key material and sends it to the server encrypted with the server's public key

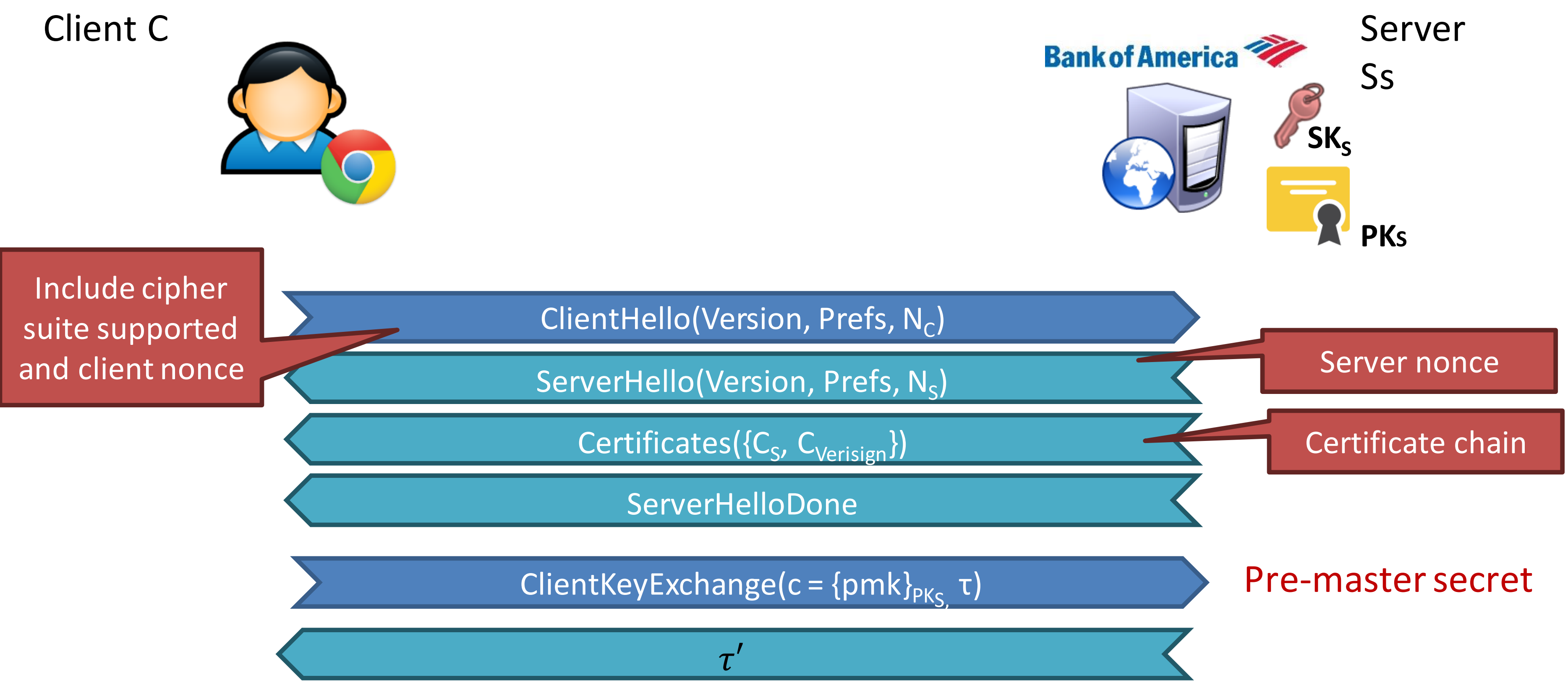
Validate server certificate



The client must validate the certificate chain to establish trust

- Does the server's DNS name match the common name in the cert?
- Are any certs in the chain expired?
- Is the CA's signature cryptographically valid?
- Is the cert of the root CA in the chain present in the client's trusted key store?
- Is any cert in the chain revoked?

TLS Handshake



Choose *pmk* random

$c \leftarrow E_{PK_S}(pmk)$

$mk \leftarrow H(pmk, N_C, N_S)$

$(k_C, k'_C, k_S, k'_S) \leftarrow PRG(mk)$

tr = all exchanged msg

$\tau \leftarrow MAC_{mk}(\mathrm{tr})$

$pmk \leftarrow D_{SK_S}(c)$

$mk \leftarrow H(pmk, N_C, N_S)$

$(k_C, k'_C, k_S, k'_S) \leftarrow PRG(mk)$

tr′ = all exchanged msg

$\tau' \leftarrow MAC_{mk}(\mathrm{tr}')$

TLS record: encryption (CBC AES-128, HMAC-SHA1)

Client: (k_C , k'_C)

Client side **Enc(k_C , k'_C ,data, ctr_C) :**

Step 1: tag ⟵ **Tag**(k'_C , [++ctr_C ll header ll data])

Step 2: pad [header ll data ll tag] to AES block size

Step 3: CBC encrypt with k_C and new random IV

Step 4: prepend header

TLS record: decryption (CBC AES-128, HMAC-SHA1)

Server side **Dec(k_C , k'_C record, ctr_C)** :

Step 1: CBC decrypt record using k_C

Step 2: check pad format: send bad_record_mac if invalid

Step 3: check tag on [$++ctr_C$ ll header ll data]

send bad_record_mac if invalid

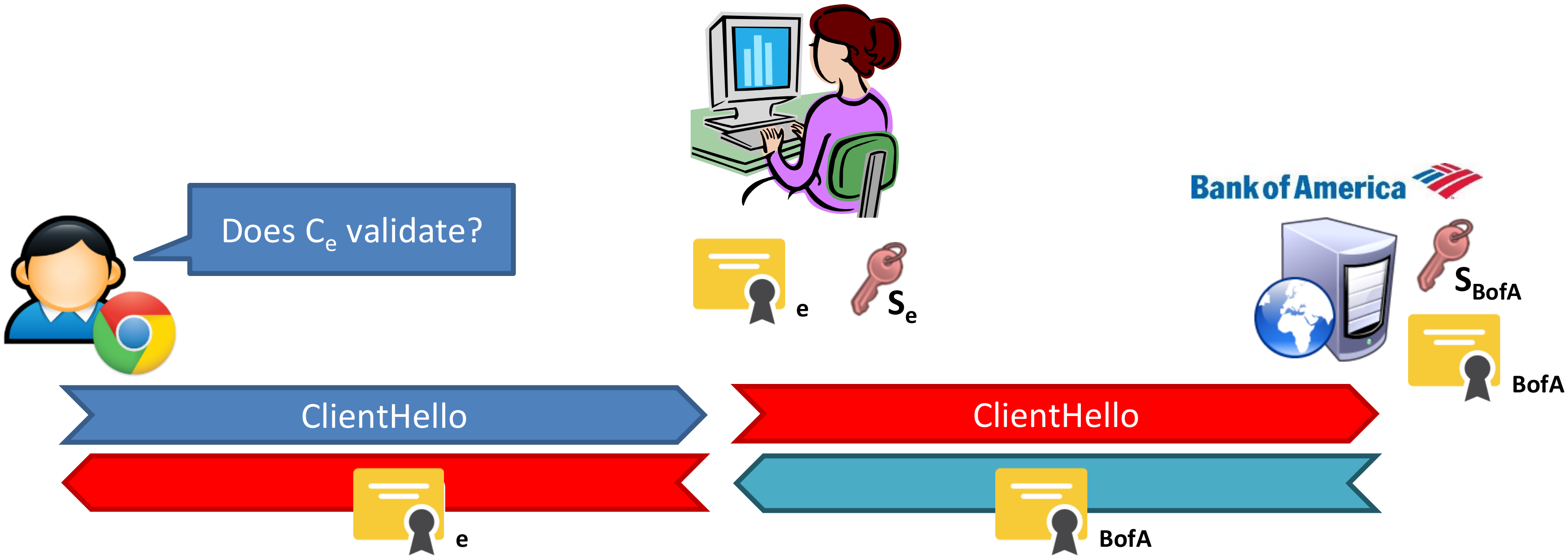
Provides authenticated encryption

(provided no other information is leaked during decryption)

Problems with TLS

- TLS is a widely deployed and extremely successful protocol
- … but its not perfect
- Problems with TLS:
  1. CA trustworthiness
  2. Weak ciphers and keys
  3. Protocol Attacks
  4. Man-in-the-middle attacks
  5. Secret key compromise
  6. Implementation Bugs

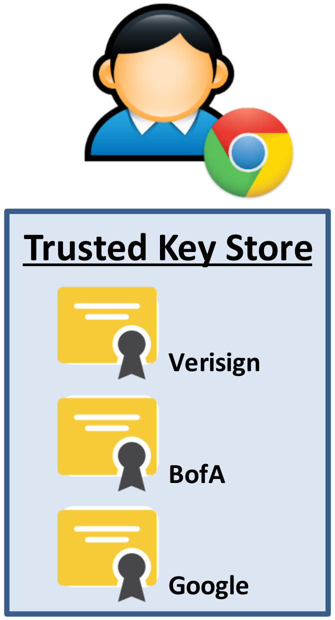
TLS Man-in-the-Middle Attack



- If C_e is self-signed, the user will be shown a warning
- If the attacker steals C_{BofA} and S_{BofA}, then this attack will succeed unless:
  1. Bank of America revokes the stolen cert
  2. The client checks to see if the cert has been revoked
- If the attacker manages to buy a valid BofA cert from a CA, then the only defense against this attack is certificate pinning

Certificate Pinning

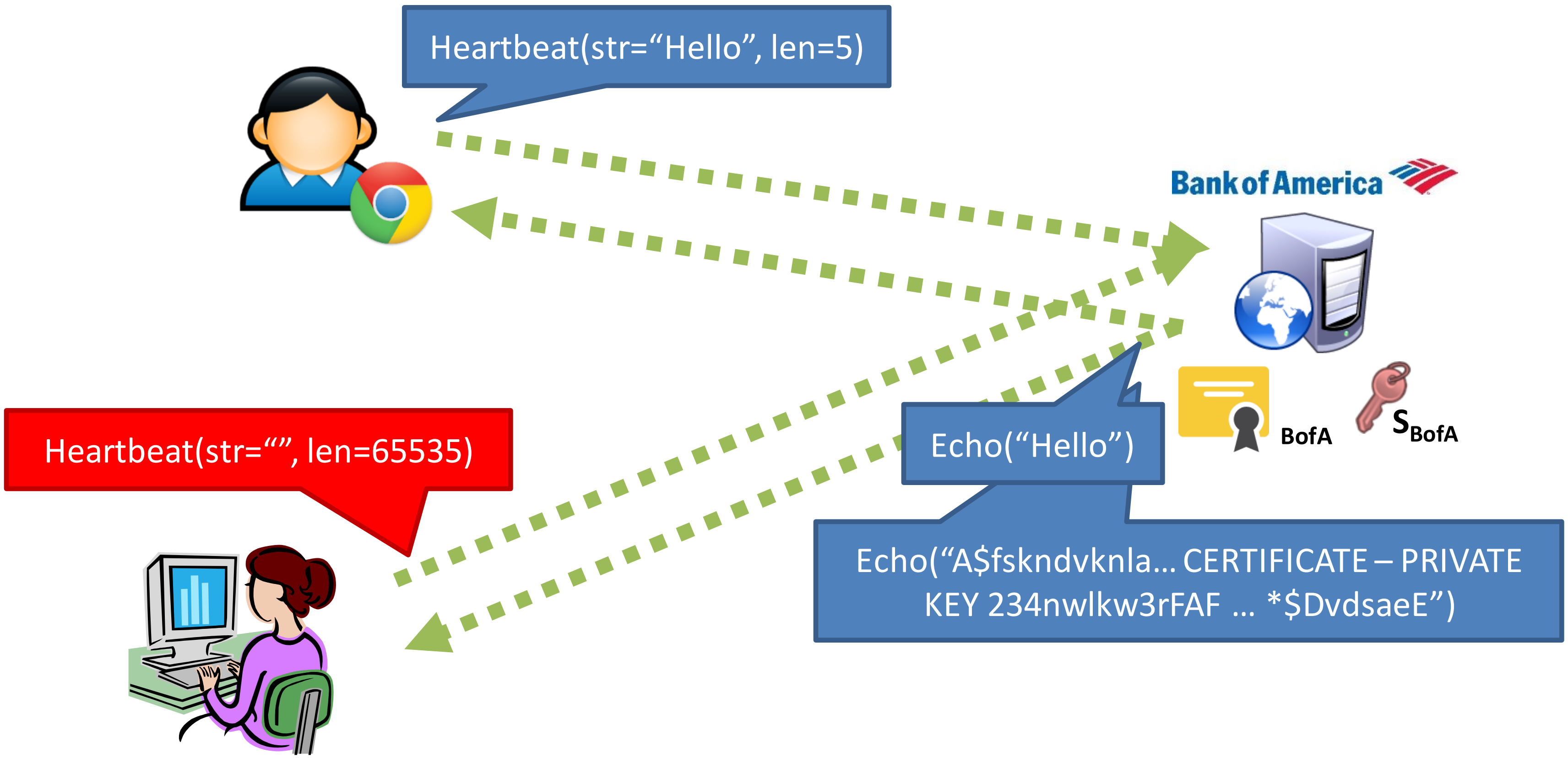
- Certificate pinning is a technique for detecting sophisticated MitM attacks
  - Browser includes certs from well-known websites in the trusted key store by default
  - Usually, only certs from root CAs are included in the trusted key store
- Example: Chrome ships with pinned copies of the *.google.com certificate
- Pinning isn't just for browsers
  - Many Android and iPhone apps now include pinned certificates
  - E.g. Facebook's apps include a pinned cert
  - Revocation done through app updates



HeartBleed

- Serious vulnerability OpenSSL versions 1.0.1 – 1.0.1f
  - Publicly revealed April 7, 2014
  - Exploits a bug in the TLS heartbeat extension
- Allows adversaries to read memory of vulnerable services
  - i.e., buffer over-read vulnerability
  - Discloses addresses, sensitive data, potentially TLS secret keys
- Major impact
  - OpenSSL is the de facto standard implementation of TLS, so used everywhere
  - Many exposed services, often on difficult-to-patch devices
  - Trivial to exploit

Heartbleed Exploit Example



Review: SSL/TLS

- Secure transport protocol used widely
  - Server authentication based on certificates
  - Handshake protocol to establish cipher method and derive symmetric keys
  - Record protocol secures all subsequent communication in a session
- HTTPS is main application on Internet
  - Secure HTTP tunnels between browser and web server
  - Transport layer secured with TLS

Bitcoin

- Digital crypto currencies
  – Advantages over paper cash
- Distributed public ledger
  – Blockchain creation and distribution
  – Proof of Work (PoW)
  – Agreement and resilience to adversaries
- Incentives for users
  – How money is created and obtained
- Bitcoin security
- Other cryptocurrencies

Bitcoin network

http://bitnodes.earn.com/

GLOBAL BITCOIN NODES DISTRIBUTION

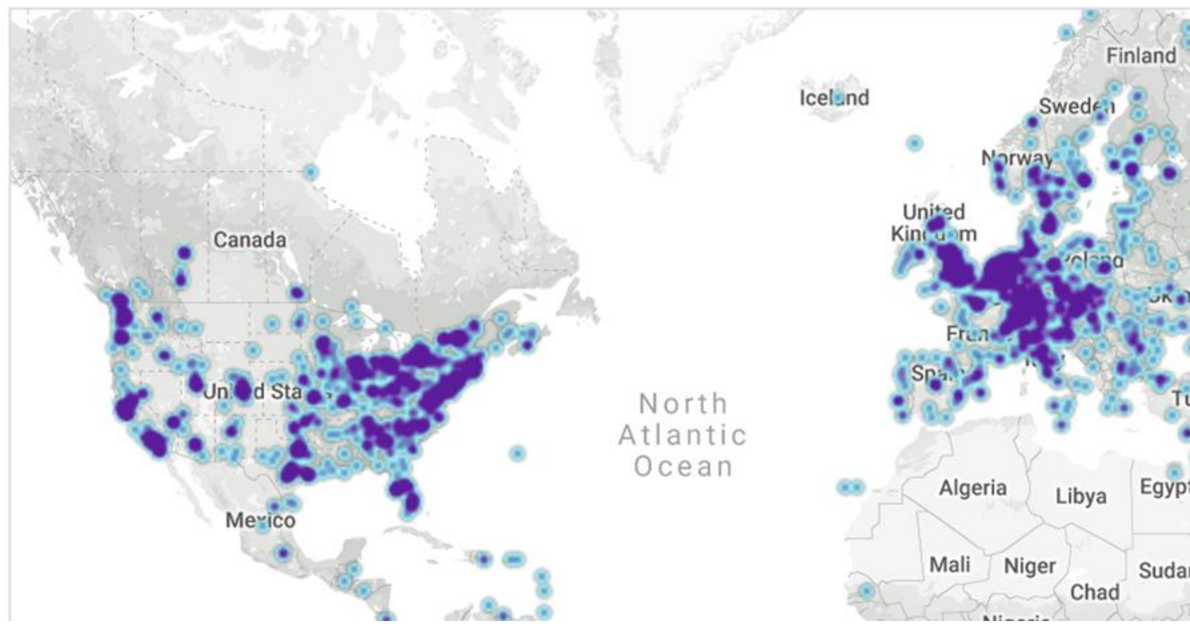
Reachable nodes as of Mon Mar 12 2018 22:56:53 GMT-0500 (Central Daylight Time).

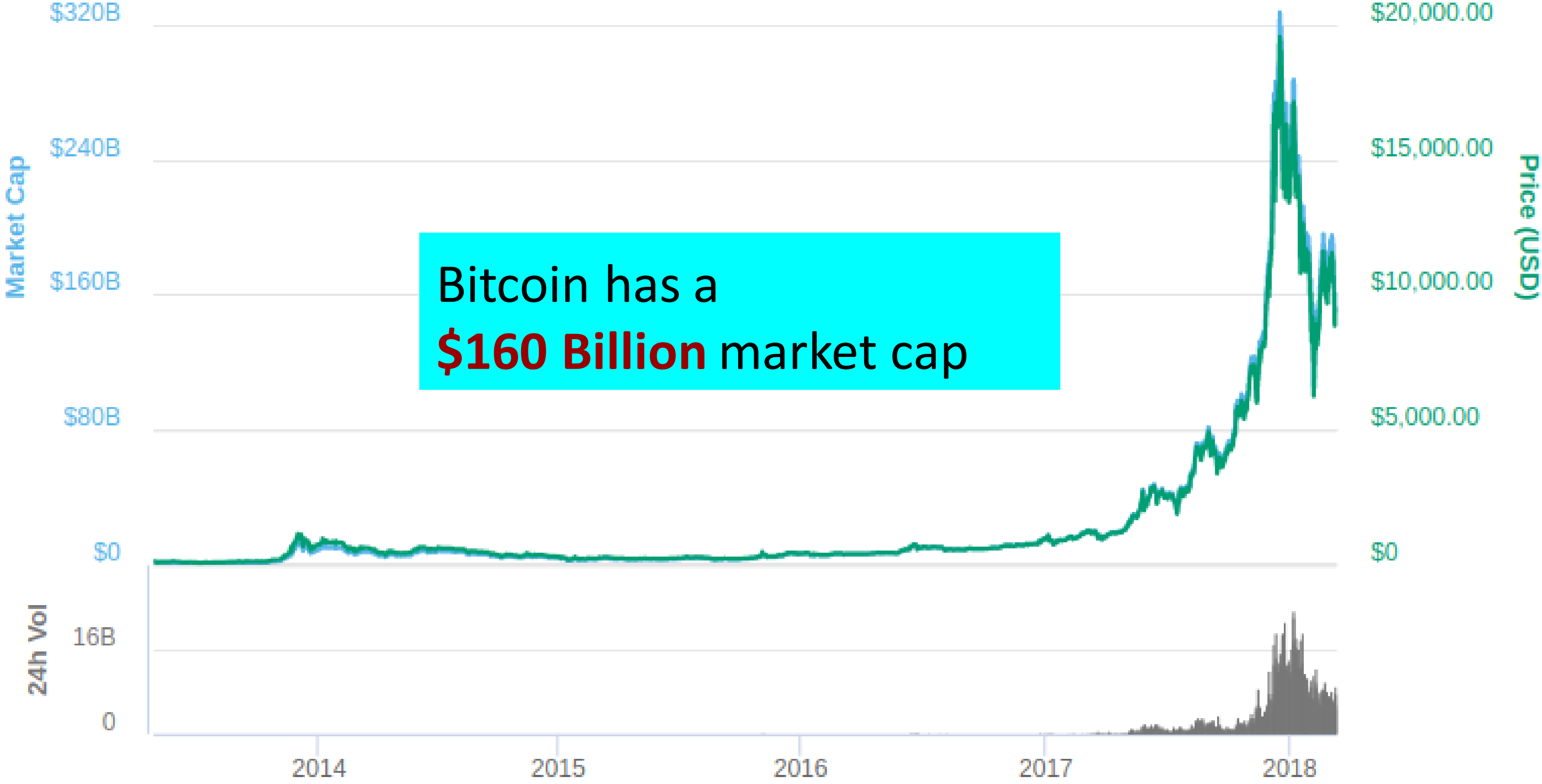
12207 NODES

24-hour charts »

Top 10 countries with their respective number of reachable nodes are as follow.

RANK	COUNTRY	NODES
1	United States	2786 (22.82%)
2	China	2245 (18.39%)
3	Germany	1962 (16.07%)





Bitcoin has a
$160 Billion market cap

Source: coinmarketcap

28

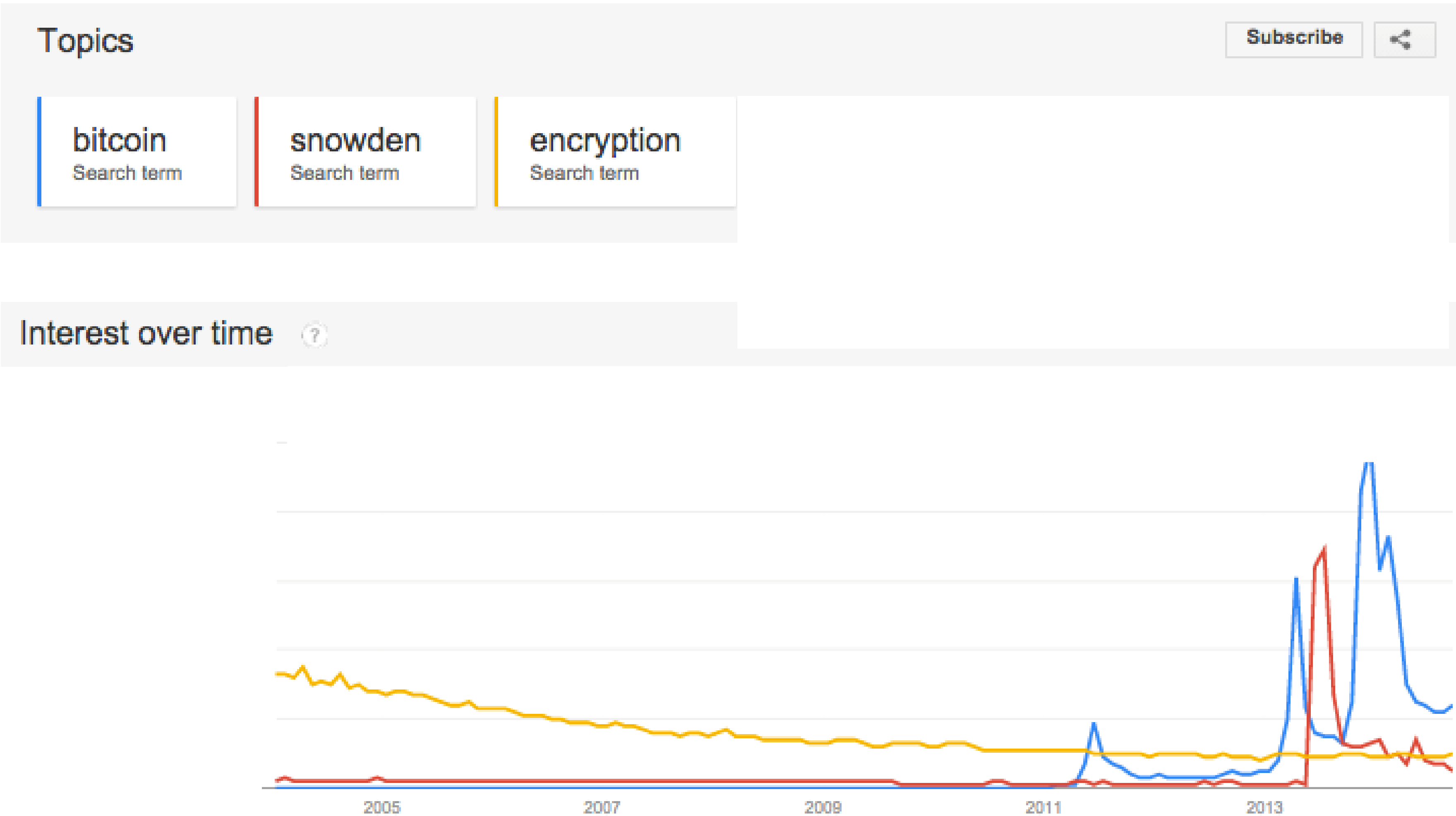
Bitcoin is the first and largest of ***hundreds*** of cryptocurrencies

All Coins Tokens

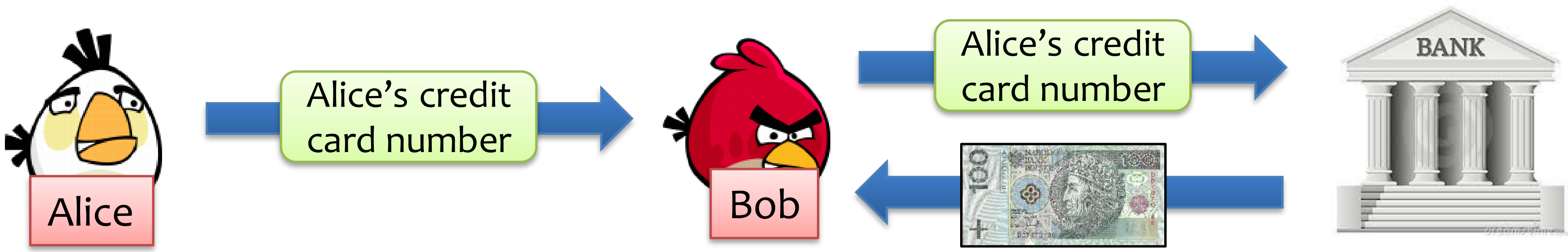
#	Name	Market Cap	Price
1	Bitcoin	$158,487,664,907	$9,369.05
2	Ethereum	$69,592,728,347	$709.01
3	Ripple	$31,329,975,058	$0.801443
4	Bitcoin Cash	$18,383,130,216	$1,080.42
5	Litecoin	$10,000,508,493	$179.85
6	Cardano	$5,858,040,099	$0.225943
7	NEO	$5,738,850,000	$88.29

https://coinmarketcap.com

Probably one of the most discussed cryptographic technologies ever!



Traditional ways of paying “digitally”



<u>BENEFITS</u>

1. Convenient (pay online)
2. Highly regulated
3. Banks handle fraud
4. Cannot double-spend
5. Tax records

<u>PROBLEMS</u>

1. **Trusted server** for each transaction
2. High **transaction fees**
3. Record of all transactions (**No anonymity/privacy)**

Bitcoin – a "digital analogue" of the paper money

A digital currency introduced by "Satoshi Nakamoto" in 2008

- First e-cash without a centralized issuing authority
  - Store and transfer value without reliance on central banks
  - Anyone can join the system and make transactions
  - Transactions are publicly verifiable
- Built on top of an unstructured P2P system
  - Participants validate transactions and mint currency
  - System works as long as the *majority of users are honest*
  - Provides economic incentives for users to be honest

Currency unit: **Bitcoin (BTC) 1 BTC = 10^8 Satoshi;** value ≈ $1250

Bitcoin

in Bitcoin:

- **No trusted server, money circulates**
- **Low fees**
- **"Pseudonymity"**

PROBLEMS WITH DIGITAL PAYMENT

1. **Trusted server** for each transaction
2. High **transaction fees**
3. **No anonymity/privacy.**

Resources

- Book: Bitcoin and Cryptocurrency Technologies
  - http://bitcoinbook.cs.princeton.edu/
- Bonneau et al. Research Perspectives and Challenges for Bitcoin and Cryptocurrencies.
  - https://eprint.iacr.org/2015/261.pdf
- Bitcoin and Cryptocurrency Technologies Course
  - https://www.coursera.org/learn/cryptocurrency
  - https://piazza.com/princeton/spring2015/btctech/resources

Acknowledgement

Some of the slides and slide contents are taken from
http://www.crypto.edu.pl/Dziembowski/teaching
and fall under the following:

We have also used slides from Prof. Dan Boneh online cryptography course at Stanford University:

http://crypto.stanford.edu/~dabo/courses/OnlineCrypto/