THE GRAINGER COLLEGE OF ENGINEERING
SIEBEL SCHOOL OF COMPUTING AND DATA SCIENCE

CS 521

Technological Foundations of Blockchain and Cryptocurrency

Grigore Rosu

Topic 2 – Basic Crypto Primitives

ILLINOIS

Thanks!

To Professors

David Tse (Stanford)

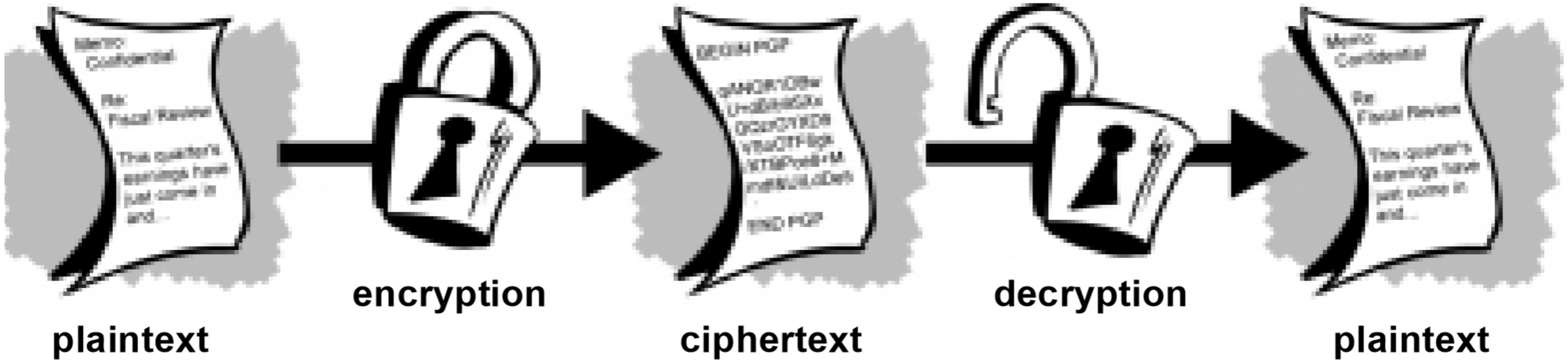
Sriram Viswanath (UT Austin)

Sreeram Kannan (UW – now at EigenLayer)

Some crypto primitives

- Encryption and Signatures
- Cryptographic Hash Functions
- Hash Accumulators
  - Blockchain
  - Merkle trees

Basic Encryption



Encryption *Decryption*

hello → lipps → hello

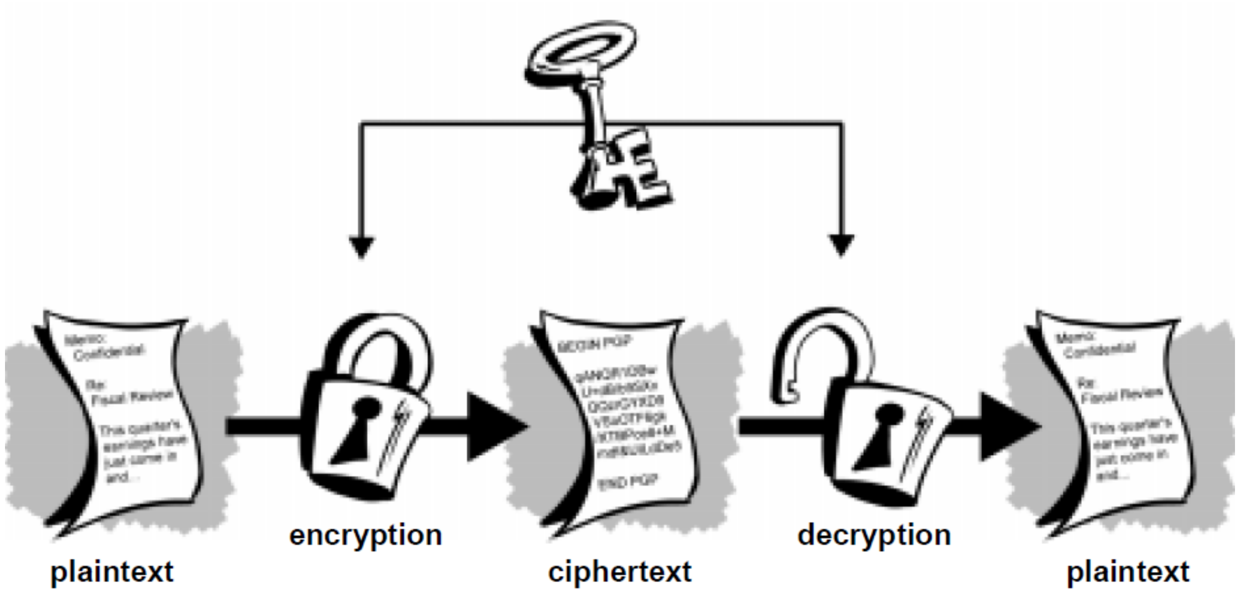
plaintext *ciphertext* *plaintext*

Cypher: Offset the Alphabet
Key: 4

Scene from “Breaking the Enigma Code”

https://youtu.be/zZuqLLdx2YQ

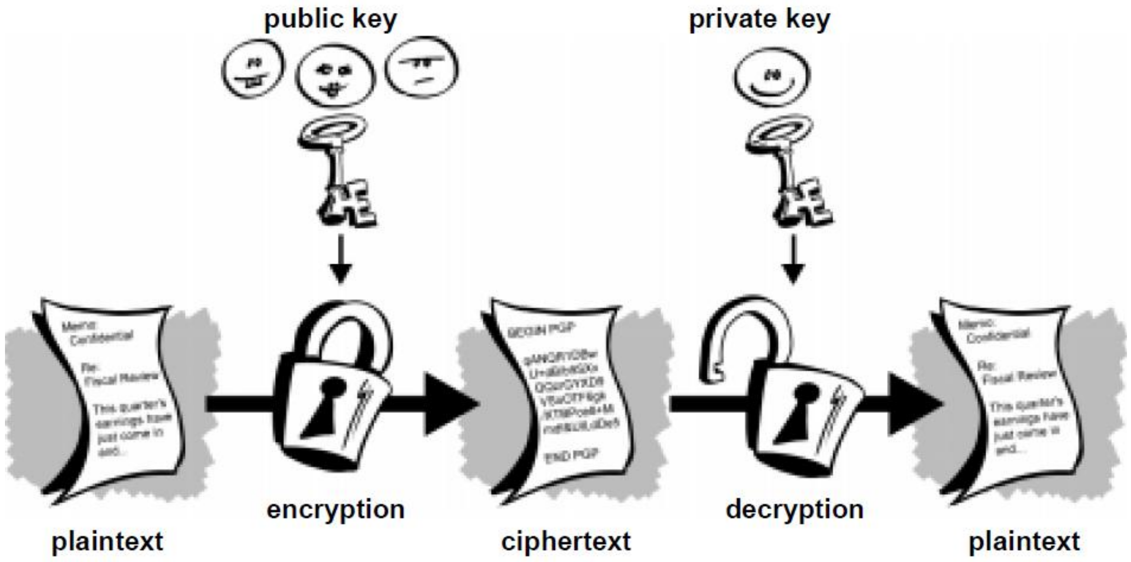
Symmetric Key Cryptography



Pros and Cons

- ✓ High performing – fast, especially if the data is not going to be transmitted
- ✓ Can be implemented in hardware and software
- ✘ Secure key distribution is difficult, requires trust and secrecy between the parties as well as trust for the "distribution mechanism" if the parties are not in the same location

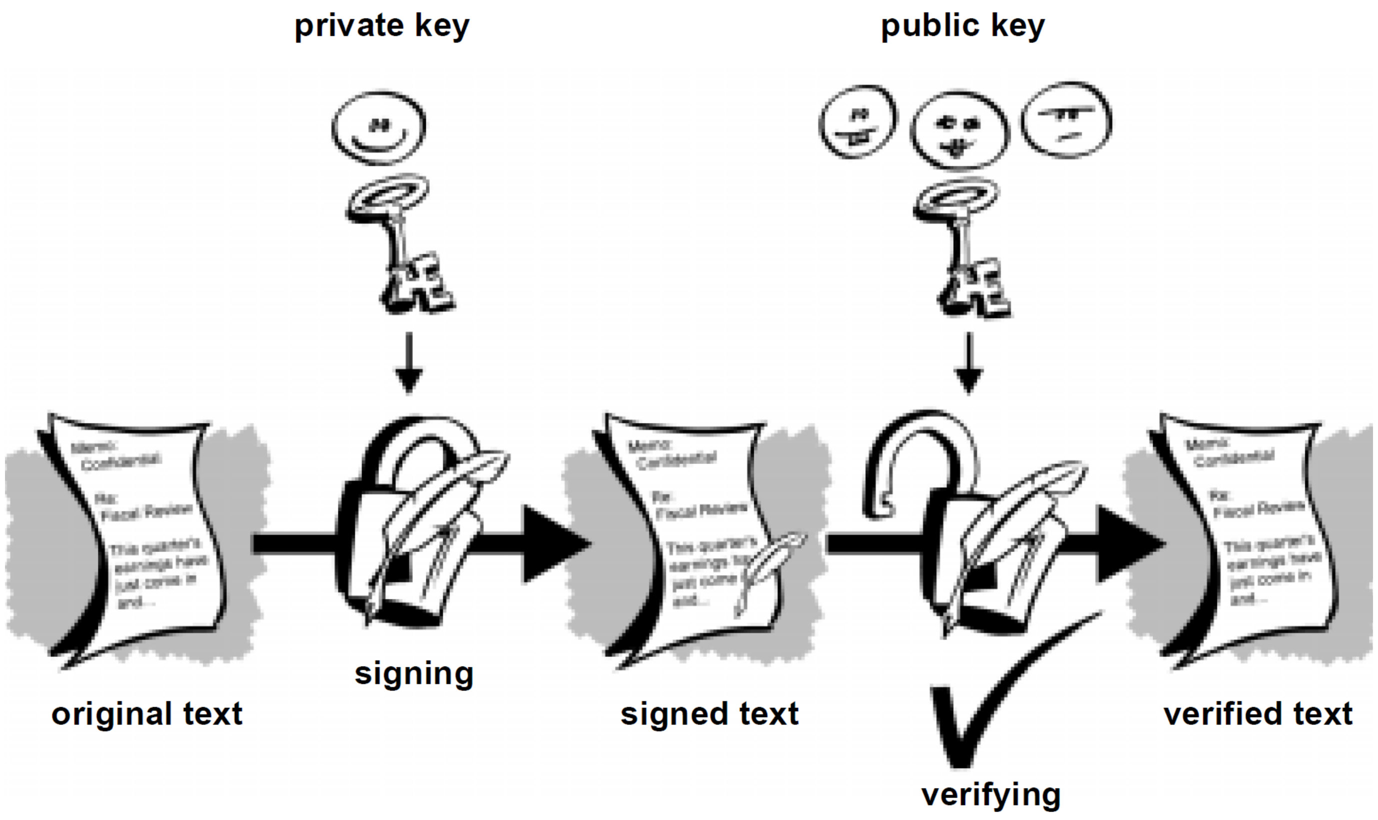
Public-Private (aka Public-Key) Cryptography



Pros and Cons

- ✓ People can exchange messages securely without a security arrangement
- ✓ Makes secure message exchange available to a wider group of people
- ✗ Does not ensure foolproof identity of the sender

Digital Signatures



Digital Signatures

- **Key generation**

`(secretkey,publickey) =`
`    Generatekeys(keysize)`

- Randomized function

- **Signature**

`Sig =`
`    sign(secretkey, message)`

- **Verification**

`verify(publickey,Sig,message)`

Unforgeable Signatures

- **Unforgeable**

Computationally hard to generate a verifiable signature without knowing the secret key

- **ECDSA**

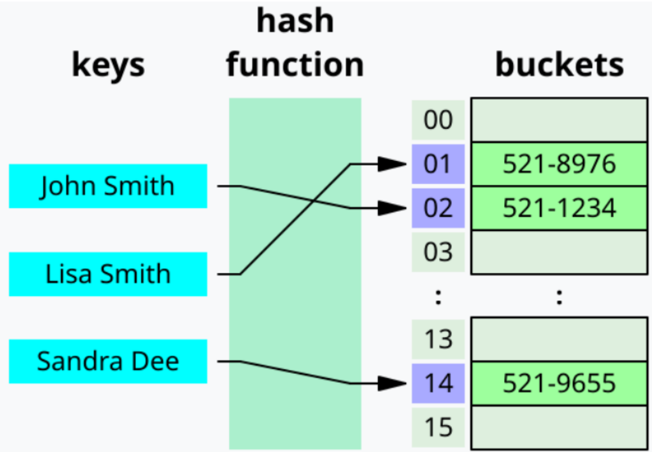
Elliptic Curve Digital Signature Algorithms

Cryptographically secure against an adaptive adversary

Decentralized Identity Management

- Public keys are your identity
  - *address* in Bitcoin/blockchain terminology
- Can create multiple identities
  - (**`publickey`**, **`secretkey`**) pairs
  - Publish **`publickey`**
  - Sign using **`secretkey`**
- Can create oneself
- Verifiable by others

Hash Functions



Defining Properties:

- Arbitrary sized inputs
- Fixed size deterministic output
- Efficiently computable
- Minimize collisions

Canonical application:

- Hash Tables
- Store and retrieve data records

Example: Hash Functions

- Division hashing

$$y = x \mod 2^{256}$$

- Uniform output
- Simple deterministic function
- Collision resistant

Cryptographic Hash Functions

Extra Properties:

- Adversarial collision resistance
  - Birthday paradox
- One way function
- Specialized one way function

Canonical applications:

- Message digest
- Commitments
- Puzzle generation
- Mining process

Hashing Algorithms

NSA 2001

No Collisions (yet)

SHA2 (Secure Hashing Algorithm)

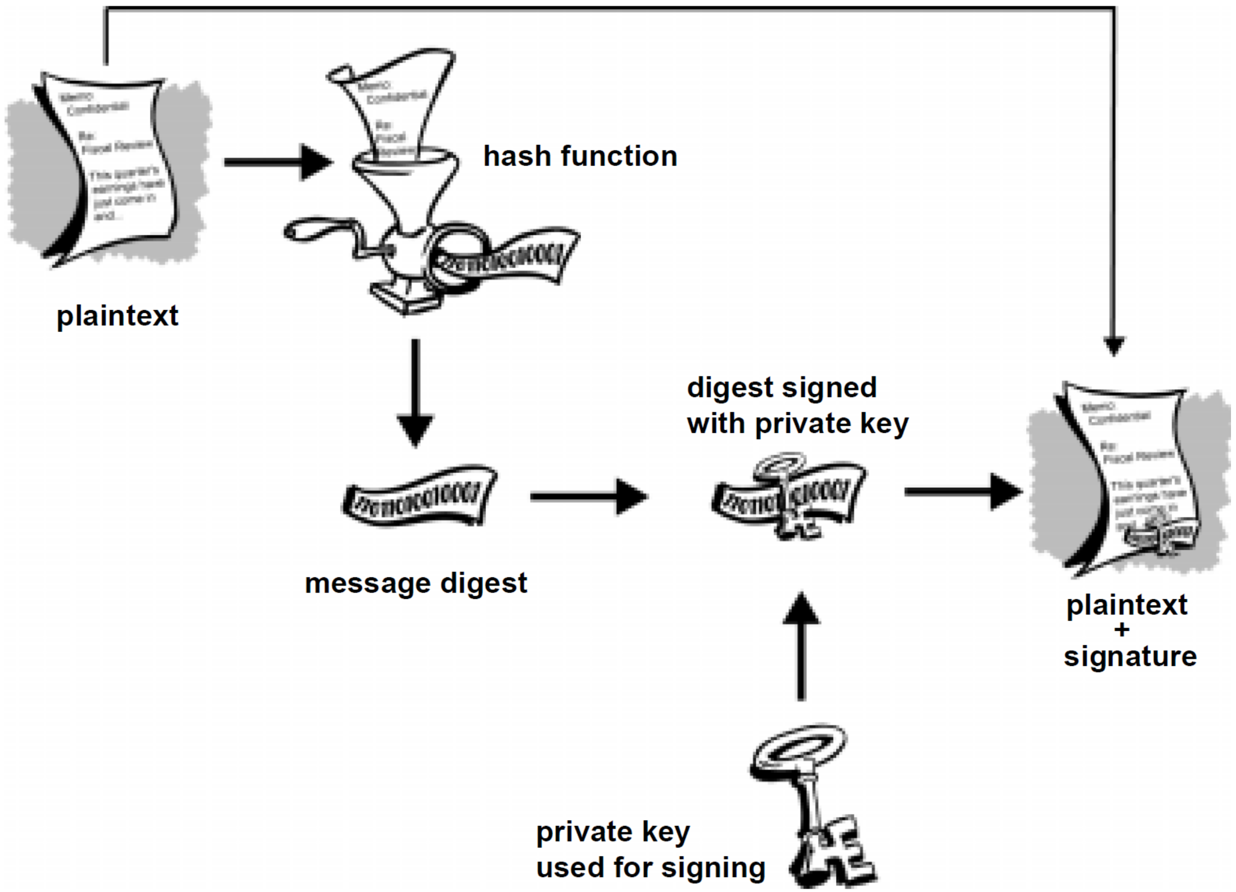
- SHA2 takes strings of arbitrary length and generates a unique and irreversible 256 (SHA256) or 512 (SHA512) bit strings (SHA2 is the successor to SHA1 that generated 160 bit strings)
- SHA1 was derived from MD4

MD5 (Message Digest)

- MD5 is also a "child" of MD4 and produces a 128 bit output string
- MD5 works by chaining a "compression function

Collisions found!

Basic building blocks together



Hash Pointer

- **Pointer to:**

  location of information

  + hash of the information
- **Regular pointer**
  – retrieve information
- **Hash pointer**
  – retrieve information and verify the information has not changed

- Regular pointers
  – Used to build data structures
    - linked lists, binary trees, etc
- Hash pointers
  – Can also be used to build data structures
  – Crucially useful for blockchains!
    - Blockchain = hash pointer based data structure

Blockchain: a linked list via hash pointers

- **Block**: Header + Data
- **Header:** hash pointer to
  location of previous block
  + hash of the previous block
- **Data**: information specific to the block (e.g., transactions)

- **Application**: tamper evident information log
- Head of the chain being known is enough to find tamper evidence in any internal block
- Hence the phrase: **block chain**

  blockchain

Merkle Tree

Binary tree of hash pointers

- Retain only the tree root
- Tamper of any data in the the bottom of the tree is evident

- **Proof of Membership**
- **Proof of Non-membership**

Merkle Trees

- **Block**: Header + Data
- **Header:** Pointer to
  location of previous block
  + hash of the previous block
- **Data**
  – block specific information