Contents

Contributors:

- Daniel Fitz (Sanchez)
- Jake Dunn (nomad)
- *Notes from UQAttic.net*

Chapter 1

Exam Structure

Exam Is Open Book!

1.0.1 What is Covered?

- All lecture content (Weeks 1, 2, 3, 5), (Required text book reading)
- All Seminars
  - For seminars that are based on a research paper (#20 - #50), both content on slides and paper is relevant
  - For all other seminars, only information presented in seminar is covered by exam
  - Emphasis in exam will be on seminars based on research papers
- Guest Lecture by Peter Robinson (W4) is not covered

1.0.2 Format

Two parts with total $\approx$ 55 marks

- Part A: 8 Questions ($\approx$25 marks)
  - Questions on Lectures
  - Material covered in lectures in weeks 1,2,3 and 5
  - Short Answer/Problem
- Part B: Answer 3 questions (30 marks)
  - Questions on Seminar Presentations
  - Mix of essay-style and short answer questions
  - Select and answer 3 out of 4 questions
  - Cannot do own seminar question, get an extra question to choose from

Part B Example Questions

- Describe what the XREP protocol presented in the paper tries to achieve, and discuss the basic mechanisms that it is using
- Further discuss for what environments it can be applied and describe its limitations and vulnerabilities
- Describe the relevance of the parameter K in the proposed protocol
- Describe at a high level what Aurasium is, and the key goals it is trying to achieve
- Describe how Aurasium interacts with the Andriod system and applications
- Describe if and how malicious application can detect the presence of Aurasium

Chapter 2

Lecture Notes

2.1 Lecture 1: Bitcoin, Cryptocurrencies Blockchain Technology

Blockchain (sometimes called Distributed Ledger) is the underlying technology on top of which Bitcoin and other cryptocurrencies are built. While cryptocurrencies such as Bitcoin have been the original application, Blockcahin technology has a much wider range of applications.

Industries to be affected by the blockchain:

- Banking and Payments
- Cyber Security
- Supply Chain Management
- Forecasting (Research, Consulting, Analysis and Forecasting)
- Networking and IoT
- Insurance (Global Trust)
- Private Transport and Ride Sharing
- Online Data Storage
- Charity
- Voting
- Government
- Public Benefits
- Health Care
- Energy Management
- Online Music
- Retail
- Real Estate
- Crowd Funding

2.1.1 Digital Cash

Properties needed in digital cash:

- Forgery (counterfeit)-proof
- Anonymity (at least to some extent)
- Cannot be spent multiple times ("Double Spend" Problem)

2.1.2 eCash

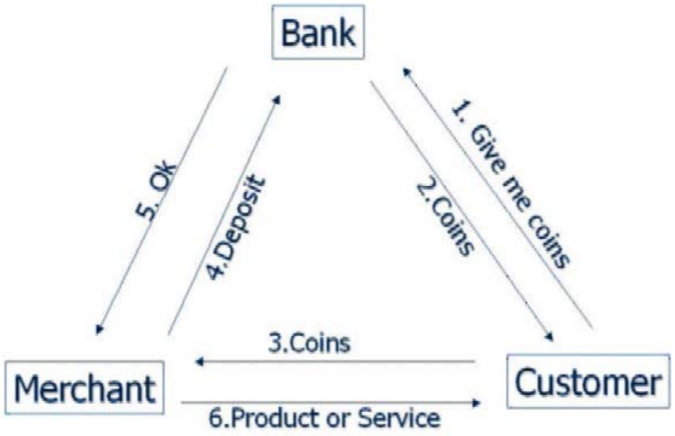


Figure 2.1: eCash Step Through

- eCash Transaction
  Each eCash coin has a unique identifer or "serial number"
- **Forgery** can be avoided by:
  - Bank digitally signing each coin
  - Creation of valid coin requires knowing private key, everybody can verify
- Avoid **Double Spending**:
  - In Step 4 (Figure), bank checks if the coin has already been spent (keeps list of serial numbers of all issued and spent coins)
- Provide **Anonymity**:
  - You might not want the bank to know what you spend your eCash on
  - Bank can trace coins through matching of serial numbers
  - **Blind Signatures**

Note 1: Blind Signatures

Signer can provide a valid digital signature without seeing what he/she is signing

How can Blind Signatures solve anonymity problem in eCash?

- Customer creates a random serial number (sufficiently large to avoid collisions)
- Sends 'blinded' serial number to bank
- Bank uses blind signature algorithm to issue a valid coin for this serial number and for the required amount, and deducts amount from user's account

- Customer can then 'unblind' the signature, and spend valid coin at merchant
- Merchant sends coin to bank for checking
- Bank checks that it is a valid coin (valid signature), and has not been spent yet, based on serial number, and adds serial number to its 'spent' list
- Important: **Bank cannot link serial number in coin to the customer to whom it was issued!**

Note 2: Bitcoin Crypto Building Blocks

- Cryptographic Hash Functions
- Hash Pointers, Blockchain
- Merkle Trees
- Proof of Work methods, "Hashcash"
- Digital Signatures

2.1.3 Cryptographic (one-way) Hash Function

- Also called: digital fingerprint, message digest
- Compact way to remember what files or blocks of data we have seen
- If two files have the same hash, we can be confident that they are the same

Properties of a hash function:

Compression: "Any size" input, fixed size output

'Easy', "efficient" to compute

One-way property: For $y = h(x)$, it is 'computationally infeasible' to find x. Also called 'preimage resistance'

Collision resistance: It is computationally infeasible to find any two distinct inputs x_1 and x_2 so that $h(x_1) = h(x_2)$ (Collisions do exist, actually there is an infinite number of them)

Ideal One-way Cryptographic Hash Function "Random Oracle" Model

- A machine with an input and an output
- When an input arrives, if the input has arrived before $\rightarrow$ give same output as last time. Else generate a new random output and record the input and output
- Best approach to get specific output is brute forcing

Note 3: Hash Pointer

- Points to a block of data (e.g. via an address), like a normal pointer
- In addition, it also stores a hash $h()$ of the data
- With $h()$ we can check the **integrity** of the data (i.e. if modified)

2.2 Lecture 2: Bitcoin, Cryptocurrencies Blockchain Technology

Blockchain is over-hyped!

2.2.1 Blockchain (or Hash Chain/List)

- Use hash pointers to build structures similar to a linked list
- Head hash pointer of the list protects the integrity of the entire list or chain (Need to store hash pointer to the head of the list externally to list)
- Hash is computed over entire block, including header, which includes hash pointer to previous block
- First block is called **Genesis Block**
- If an attacker modifies a block, they need to modify the contents of all subsequent blocks

Merkle Trees

Note 4: Merkle Trees

- Named after Ralph Merkle
- Can protect integrity of large number of data blocks, like a Blockchain
- We only need the **Root Hash** at the root of the tree (**Merkle Root**)
- Modification of any data block by attack results in different hashes all the way up to the Merkle Root, and can easily be detected
- Cost to prove `Tx1` in the tree: $O(\log_2 N)$

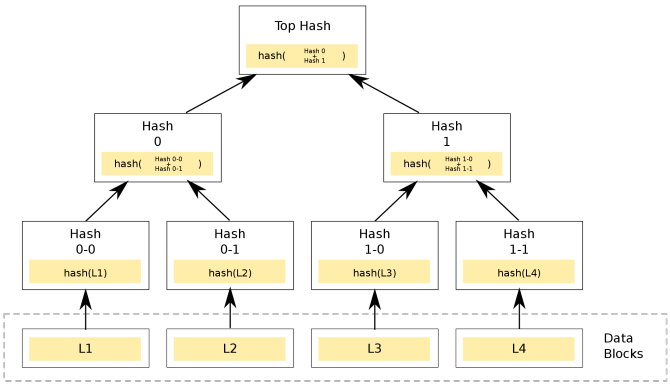


Figure 2.2: Merkle Tree Layout

Use of Merkle Trees in Bitcoin

Bitcoin uses Merkle Trees to store Transactions in a "Block" (≈ 2000)

- Each Block stores a **Merkle Root**
- Merkle Trees allow verification that a transaction is part of a Block without having the entire block, only Merkle Path is required. This is used to implement Simplified Payment Verification (SPV) in Bitcoin

Blocks are then stored in a Blockchain

2.2.2 Public Key Cryptography – Digital Signatures

Properties of signatures

- Only **you** can provide a valid signature, anyone can verify
- Signature is tied to a particular document, cannot be copy-pasted to another document

Public Key Cryptography

- Asymmetric operation
- Two keys: Public key and Private key

Encryption

- Encryption with Public Key
- Decryption with Private Key
- Key Benefit: Simplified key distribution/management
- Remaining security problems $\rightarrow$ Authenticity of public key, Public Key Certificates (map public key to identity)

Digital Signature

- Sign with Private Key
- Verification with Public Key

> Since public key operations are computationally expensive, digital signatures are typically applied to a hash, rather than entire file or block of data

Digital Signatures in Bitcoin

Bitcoin transactions have digital signatures

- Signed by the owner(s) of the source funds (Bitcoin to be transferred)
- This proves ownership of funds
- Prevents forgery of coins/transactions

Note 5: Bitcoin Identity

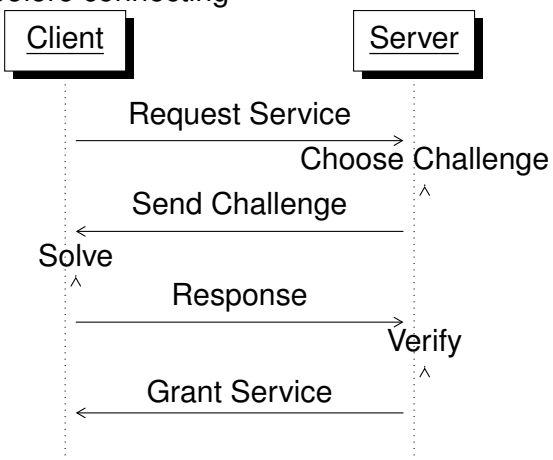
An identity in Bitcoin (a **Bitcoin Address**) is simply a public key (160-bit hash of it, to be precise)

- No need for public key certificates
- No need to link public key to a real name

Hashcash

Note 6: Hashcash

Prevent of mitigate **denial-of-service** (DoS) attacks by requiring the sender to solve a puzzle before connecting



- Require the first n bits of $h(x)$ to have a given value, first n bits are all 0 (partial pre-image)
- Same as saying $h(x) < T$
- Best approach is brute forcing
- Chance of guessing on single try: 2^{-n}, expected number of tries until success: 2^n

Bitcoin aims to have a block solved roughly every 10 minutes, difficulty is adjusts every 2016 blocks (≈ 2 weeks)

2.2.3 Cryptocurrency

Broadcasting of transactions: Unstructured P2P Network, flooding (as used in Bitcoin)

Avoiding forgery (transactions, coins): Digital Signatures

Maintaining the public ledger: P2P Network, Proof-of-work and Incentive mechanism

2.3 Lecture 3: Bitcoin, Cryptocurrencies Blockchain Technology

2.3.1 Sybil Attack

- The Sybil attack in computer security is an attack wherein a reputation system is subverted by forging identities in peer-to-peer networks.
- Creates a large number of false identities in order to flood the network and take it over.
- Named after the subject of the book **Sybil**, a case study of a woman diagnosed with multiple personality disorder
- Such false identities are called **"Sybils"**, in the context of P2P systems
- One approach to preventing these âĂIJSybil attacksâĂİ is to have a trusted agency certify identities.
- Or **"Sockpuppets"** in the context of the Internet, e.g. to manipulate public opinion

> "... referred to a false identity assumed by a member of an Internet community who spoke to, or about, themselves while pretending to be another person. 'The term now includes other misleading uses of online identities, such as those created to praise, defend or support a person or organization, to manipulate public opinion, or to circumvent a suspension or ban from a website"
> - Wikipedia

2.3.2 Decentralized Consensus in Bitcoin

- Bitcoin achieves consensus by replacing **one node (or one IP address) one vote**, with **one CPU one vote**
- Bitcoin mining uses mostly ASICs and GPUs, not CPUs
- Consensus is achieved by a (probabilistic) majority vote, based on computing power
  - Attacker needs > 50% of combined computing power in order to cheat with high probability
  - This is harder to achieve than controlling more than half of the nodes
  - It's relatively easy to spin up a few thousand VMs as nodes, compared to controlling >50% of Bitcoin's mining compute power
- Bitcoin combines a **proof-of-work mechanism** (based on Hashcash), combined with a clever incentive mechanism $\rightarrow$ Nodes get paid to do the right thing, i.e. checking and confirming valid transactions via "mining"

2.4 Lecture 3: Bitcoin

- Complete transaction history is stored in public ledger (blockchain), stored by nodes in P2P network
- New transactions are created off-line, and then broadcast is P2P network
- Nodes validate and relay transactions (if valid)
- Nodes add new transactions (not part of blockchain yet) to a **transaction pool**
- Nodes combine transactions in pool to a block, and try to solve the corresponding proof-of-work puzzle, to "confirm" the block
- Node that finds solution first broadcasts block with solution in the network (Winning node selection is probabilistic, with probability propertional to computing power)
- Nodes check solution, and if OK, add new block to their local copy of blockchain
- All nodes who were working on solving the old puzzle, immediately start working on a new block
  - Everybody always works on the longest chain
  - Convergence to longest chain provides consensus
  - Forks happen but are resolved by (computational) majority vote

Note 7: Bitcoin Components

- Bitcoin Network
- Bitcoin Identities/Addresses
- Bitcoin Transactions
- Bitcoin Scripting Language
- Blocks
- Proof-of-work, Hash puzzles, Mining
- Consensus

2.4.1 Bitcoin Network

Transactions are broadcast in the Bitcoin network, consisting of "full" Bitcoin nodes ($\approx$10,000)

- Best-effort (asynchronous, unreliable), it's enough if only some nodes get the message

- It's easy to join the Bitcoin Network, just download and run the client (Bitcoin is a **permissionless, public** blockchain, anyone can join)

Network is unstructured P2P network (random topology)

- Similar to Gnutella (a P2P system from a long time ago)
- Overlay network, TCP, port 8333
- Messages are flooded
- All nodes are equal, no hierarchy
- Nodes can join at any time
- Node is 'forgotten' if it does not respond for more than 3 hours
- Network is very simple and robust, e.g. to churn, but not very efficient

Note 8: Bitcoin Network Nodes

- Check validity of transactions
- Relay transactions in the network via flooding
- Mining (proof-of-work puzzles)
- Validate and forward confirmed blocks (Add them to local copy of blockchain)

2.4.2 Bitcoin Identity and Addresses

Users are represented by their Public Key addresses (hash), called **Bitcoin Addresses**, which serve as a pseudonyms

- An address is a **160 bit** value, and is computed as follows
- RIPEMD160 hash of SHA-256 hash of ECDSA Public Key
- This is a "Pay-to-pubkey-hash (P2PKH)" address, encoding starts with "1"

When spending a coin, spender needs to proof ownership of coin by providing a valid digital signature on the spend transaction (i.e. proving ownership of corresponding private key)

How can digital signatures be verified?

- We need public key, not just hash of public key
- Spender of coin needs to provide both valid signature AND public key
- How do we know the provided public key is authentic $\rightarrow$ Hash it, and compare with Bitcoin address, which is the hash of public key

Bitcoin also supports **Pay to script hash (P2SH)** addresses

- Allow transactions to be sent to a **script hash** instead of a public key hash (Address encoding starts with a '3' instead of '1')
- To spend bitcoins sent via P2SH, the recipient must provide a script matching the script hash and data which makes the script evaluate to true
- Allows more complex transactions (**smart contracts**), e.g. transaction outputs that require multiple signatures (multisig), or transaction puzzle, ...

Note 9: Bitcoin Address Encoding

Bitcoin uses Base58 encoding (binary-to-text encoding). Similar to Base64 encoding, but without some characters. Rationale, as explained in original bitcoin client source code:

```
// Why base-58 instead of standard base-64 encoding?
// - Don't want 0OIL characters that look like the same in some fonts and
// could be used to create visually identical looking account numbers.
// - A string with non-alphanumeric characters is not as easily accepted as an account nbr.
// - E-mail usually won't line-break if there's no punctuation to break at.
// - Doubleclicking selects the whole number as one word if it's all alphanumeric.
```

Bitcoin also adds 4 byte checksum to addresses

Bitcoin Address – Balance

- The "balance" of an address is the total of unspent transaction outputs (UTXO) sent to the address.
- "There are no accounts or balances in bitcoin; there are only unspent transaction outputs (UTXO) scattered in the blockchain"
- A user typically has many different addresses, all managed by the "wallet" software
- The wallet "balance" is the sum of all unspent transaction outputs of all addresses owned by the user

2.4.3 Bitcoin Transactions

- Transaction are created off-line (No need to be connected to Bitcoin network for this)
- Transactions are broadcast in Bitcoin P2P network

- Nodes check validity, and relay transaction (flooding)
- Nodes add to new transactions to a block and try to solve hash puzzle
- A block with a solved puzzle is “confirmed”, and broadcast in the network, and added to the blockchain of each node
- Contains:
  - Inputs (any number ≥ 0)
  - Outputs (any number > 0)
  - Digital signatures of input coin owners (Typically for **P2PKH** transactions)
  - Input needs to be completely consumed (With exception of Transaction Fee)

Basic Transaction Types

- Common Transaction
  - 1 input
  - 1 “normal” output
  - 1 change output (back to owner)
    Create new “Change Address” to maintain “anonymity”
- Aggregating Transaction
  - Multiple inputs
  - 1 output
- Distributing Transaction
  - 1 input
  - Multiple outputs
- “Coinbase Transaction”
  - 0 input, 1 outputs
  - Freshly created (“minted”) coins
  - Miner gets this as a reward for solving Hash puzzle (and thereby confirming block)
  - First transaction in every block

2.4.4 Bitcoin Scripts

- Two types of Bitcoin scripts to validate transactions
  - a locking script (Typically **ScriptPubKey**)
  - and an unlocking script (Typically **ScriptSig**)
- A locking script is a condition placed on an output
- It specifies the conditions that must be met to spend or consume the output in the future
  - Typically a **valid digital signature** of the claimed owner
  - Can be other things, to implement basic ‘smart contracts’

Bitcoin Scripting Language “Script”

- Allows to program conditions required for the spending of Bitcoins (“Programmable Money”)
- **Script** is a simple, stack based language
- Not Turing complete (e.g. no loops)
- Scripts are guaranteed to terminate after a fixed number of steps, e.g. no infinite loops
- Why is this a good thing?
  - Avoids potential denial of service attacks on nodes, “logic bombs”
  - Remember: Every node runs all scripts to validate all transactions (BTW, this severely limits scalability of Bitcoin)
- In contrast, Ethereum has a Turing complete scripting language
  Solves DoS attack problem by putting a price on script computation (‘Gas’)

2.4.5 Blocks

- Transactions are grouped into blocks
  This is an optimization. Confirming individual transactions and adding them to the blockchain would be possible, but very inefficient
- Transactions are stored in a Merkle Tree, with the Merkle Root (Tx_Root) stored in the block header
- Once confirmed (via solving hash puzzle), a block is added to the blockchain
- The ‘Nonce’ is the solution of the hash puzzle

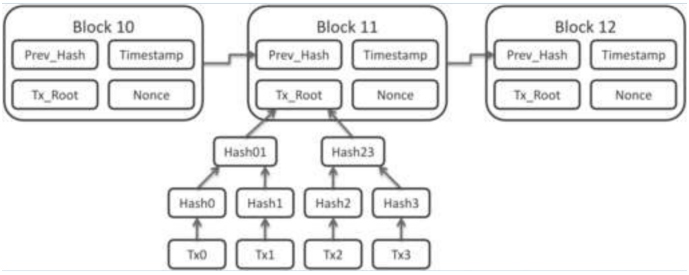


Figure 2.3: Block Structure

How are blocks added to the Blockchain?

- Multiple nodes (miners) are working towards solving the hash puzzle for a new block
  miners are probably working on different version of blocks, depending on content on their transaction pool
- First node who solves puzzle, broadcasts new block with solution (nonce) in the Bitcoin P2P network

  - Choice of winning node is random (probabilistic)
  - Probability of success is proportional to computing power of miner
- "Block Height" is sequence number of blocks
- Other nodes check if solution is correct, and if so, add block to their local copy of the blockchain, and forward new block to other nodes

2.4.6 Bitcoin Mining

- At the current level of difficulty, solving Bitcoin hash puzzles is hard and expensive
- Mostly ASICs based, some GPUs
- Total hash rate of entire Bitcoin network $\approx 25 \times 10^{18}$H/s
- Hash rate of an Intel i7 CPU ≈ 10MH/s

  Need > 10,000,000 laptops to be able to mine one block per year on average
- Finite amount of BTC ≈21 Million
- Bitcoin is deflationary (value increases, people are potentially hoarding Bitcoin)

Mining Incentive/Reward

- Transaction Fees
  - Difference between total input and total output value of transaction
  - Optional, but miners prioritize inclusion of transactions into blocks based on fees (Like giving a tip)
- Block Reward
  - For each solved puzzle (confirmation of block), miner currently gets freshly minted 12.5 BTC
  - "Coinbase transaction" (1^{st} transaction in each block)
  - This is the only way in which coins are generated in Bitcoin
  - Reward halved every 4 years (initially 50 BTC)
  - Essentially no more Block Reward in 2040

2.4.7 Consensus

- Blockchain forks can happen, e.g. due to race conditions and variable latency in Bitcoin network (*two nodes might find a solution to the puzzle at almost the same time*), or due to double spend attempt
- Nodes converge on one chain, they all aim to work on the longest (main) chain, and eventually one chain wins
- Incentive to work on the longest chain (Block Reward and TX fees can only be spent if block remains in the longest chain (need a certain number of **confirmations**, i.e. blocks added))
- Block on discontinued branches are called "orphaned blocks"
- Heuristic

  Transactions are considered "confirmed" if they are in a block that has at least 6 blocks added to the chain (i.e. 6 confirmation) *(There is nothing special about the number 6)*
- This means if someone wanted to reverse a transaction, they would have to go back 6 blocks, and re-do all the hash puzzles, before another nodes adds a new block at the end of chain (Practically impossible, unless attacker hash more than 50% of computing power of entire network)
- Other types of attacks:
  - Publish an invalid transaction, e.g. trying to spend coins that he does not own (no valid signature). Honest nodes would reject the transaction
  - Double spend attack. Cause a fork in the chain, majority vote will guarantee that only one chain will exist
  - Launch DoS attack against
- With more than 50% computing power, more beneficial to play by the rules and gain Block Rewards

2.5 Lecture 5: Secret Sharing

Requirements:

- Neither Alice nor Bob alone should be able to reconstruct M
- If Alice and Bob cooperate, they should be able to reconstruct M
- Choose a random m-bit number $X1$
- Calculate $X2 = M \oplus X1$ ($\oplus$ is bit-wise XOR)
- Give $X1$ to Alice and $X2$ to Bob
- Construct the message $M = X1 \oplus X2$

2.5.1 One Time Pad (OTP)

- The only unbreakable (classical) cipher ($C = M \oplus K$, $M = C \oplus K$)
- "Perfect security or secrecy"

  - Cannot be broken, even by attacker with infinite amount of computing power and time
  - This is in contrast to "computational security" (e.g. it would take an attacker > 1 Billion years to brute force, with all the computing power currently available)
- Not practical
  - Need very large key (same size as plaintext)
  - Key needs to be completely random
  - Key cannot be reused

Secret Sharing among 3 people

- We want: $M = X1 \oplus X2 \oplus X3$
- X1 and X2 are chosen randomly, $X3 = X1 \oplus X2 \oplus M$
- If there are only two shares, it still produces a random number

Note 10: Secret Sharing among n people

- Split secret M between n people
- Generate $n-1$ random numbers
- $X_n = X_1 \oplus X_2 \oplus X_3 \oplus \dots X_{n-1} \oplus M$
- $M = X_1 \oplus X_2 \oplus X_3 \oplus \dots X_{n-1} \oplus M_n$
- Knowing fewer than n shares gives absolutely no information about the secret M

2.5.2 Threshold Secret Sharing

- (t, n)-threshold scheme
- Method of sharing secret M among set of n participants, only t participants needed to reconstruct M

Shamir's Threshold Secret Sharing

- Invented by Adi Shamir ('S' in RSA) in 1979
- Linear Equation: $y = mx + c$
  m: random number (kept secret)
  c: secret to be shared
- Distribute shares of the secret by points on the line. (X=0, Y) not a good option
- 2 points uniquely determine a straight line and hence get the message
- We cannot loose precision (round), otherwise we loose the ability to completely reconstruct the secret (use integers only)
  Do computations in a **Finite Field**

Note 11: Shamir's (3,n)-threshold

- Use a second order polynomial
- $y = a_2x^2 + a_1x + c$
  c: secret
  a_1, a_2: random (kept secret)

Lagrange Interpolation

- Use Lagrange Interpolation to solve the equation
- Given t points (x_j, y_j), Lagrange Interpolation allows finding a polynomial $P(x)$ of degree $(t-1)$ that goes through these points
- Shamir's secret sharing scheme is sometimes called "Lagrange Interpolation Scheme"

$$P(x) = \sum_{j=1}^{t} P_j(x)$$

$$P_j(x) = y_j \prod_{k=1, k\neq j}^{t} \frac{x - x_k}{x_j - x_k}$$

Note 12: Fields

> "Algebraic structure where we can 'add' and 'multiply' they way we are used to with 'ordinary numbers'

Properties:
- Commutativity $a+b = b+a, a \times b = b \times a$
- Distributivity $a \times (b+c) = (a \times b) + (a \times c)$
- Associativity $a + (b + c) = (a + b) + c$, $a \times (b \times c) = (a \times b) \times c$
- Closure: For any a, b belonging to F, $a+b$ and $a \times b$ also belong to F
- Neutral elements
  Addition $a + 0 = a$
  Multiplication $a \times 1 = a$
- Inverse elements
  Addition: For every a in F, there exists an element $-a$ in F, such that $a + (-a) = 0$
  Multiplication: For every $a \neq 0$ in F, there exists an element a^{-1} in F, such that $a \times a^{-1} = 1$

Finite Field (Galois Field)

- Finite Field (Galois Field): Field with finite number of elements ('numbers')
- For a prime number p, integer addition and multiplication **modulo** p is a Finite Field!

- Extended Euclidean Algorithm can be used for computing large multiplicative and inverses of mod numbers

2.5.3 Research Challenges

- How to avoid cheating
- How to reconstruct secret without having to reveal shares
- How to verify that all participants provide a valid share (*Verifiably Secure Secret Sharing*)
- How to refresh shares (*Proactive Secret Sharing*)

2.5.4 Threshold Cryptography

- Shamir's scheme provides the basic concept for **Threshold Cryptography**
- Threshold cryptography has a range of applications
  - It can be used whenever we cannot or do not want to rely on a single entity (person) to perform some cryptographic operation during the operation of the system (P2P applications, wireless ad-hoc networks, Blockchain, ...)
- Threshold signature schemes
  - Instead of relying on a single entity to provide a digital signature, this task can be distributed among n entities, using an (t, n)-threshold scheme
  - Only t out of n participant need to sign a document with their partial signature to create a valid signature

Chapter 3

Seminar Slides

3.1 Ethereum: Sharding

- In all blockchain protocols currently, each node stores all the states and processes all the transactions. Good for security but terrible for scalability
- Sharding tries to solve this by allowing smaller subsets of nodes to verify transactions
- This concept is borrowed from traditional relational databases
- Each shard gets its own set of validators (proof-of-stake is required for sharding blockchains)
- Sharding is generally considered a good idea because it increases scalability without sacrificing decentralization and without reducing the security of the blockchain
- Attackers need to control approximately 33% of the validator pool to have a chance at taking over the entire blockchain to successfully attack a shard
- Random sampling bolsters the security of the system, each shard, making it hard for attackers to coordinate attacks because they do not know which shard the hostile nodes will end up in

3.2 Proof of Stake

- Proof-of-stake improves upon proof-of-work by providing greater security, reduced risk of centralization and drastic improvements to energy efficiency
- In PoS, the creator of the next block is decided randomly, and requires candidates to raise a 'stake' of the cryptocurrency in order to be considered
- This stake is known as a bond, and it is used as a collateral to vouch for a block
- In PoW you know a chain with the highest collateral (higher value of bonds)
- An idea present in this implementation is that people who want to create the next block are invested into the platform and the cryptocurrency, as they are providing a stake of their commitment. In proof-of-work, a miner only has to solve the puzzle
- The 'Nothing at Stake' problem is an issue initially faced in Peercoin, where there was only rewards given for creating blocks, and no punishments. This creates the situation where a validator, in the case of a forked blockchain, is incentivized to build upon both of the forks, rather than one or the other (which would be the case in proof-of-work, because the miner has to commit to solving the puzzle for a particular fork, a miner does not have the luxury of choosing)
- A solution to the 'Nothing at Stake' problem is to punish a validator for creating a block on two separate chains, by deducting the value in their stake, essentially fining them for bad behavior
- The 'Long-Range Attack' vulnerability is another issue with Proof of Stake
- The long range attack can be done if there is an attacker with 1% of all coins at or shortly after the genesis block. that attacker then starts their own chain, and starts mining it. Although the attacker will find themselves selected for producing a block only 1% of the time, they can easily produce 100 times as many blocks, and simply create a longer blockchain in that way

3.3 Solidity: Ethereum's Smart Contract Language

- Statically-typed programming language
- Compiles to bytecode which is run on the EVM
- Every operation (OP_CODE) in the bytecode costs gas, this makes sure that smart contracts cannot run forever on the Turing Complete EVM
- A smart contract, in the case of Solidity, is a collection of code (the functions of the smart contract) and data (the state of the smart contract), which resides at a specific address on the Ethereum blockchain

3.4 IOTA

- Runs on a non-blockchain distributed ledger, designed for IoT devices
- Features
  - Designed to support micropayments as no transaction fees exist
  - Trinary based system
  - No new IOTA's every produced, all created in genesis node
- Mining
  - Users are the miners, every user must perform work to provide the security/integrity of the network, because of this no miner fees needed
- Distributed Ledger
  - IOTA's key difference is that it is not implemented on a Blockchain, but implemented on a "Tangle", which is just a DAG (Directed Acyclic Graph)
  - Biggest benefit of the DAG system is that it's more scalable, and should actually perform better with more nodes contributing to the Tangle
- Address Creation
  - Uses keccak-384 to produce public and private keys
  - Public Key addresses should only be used once to send, as sending exposes private key by using Winternitz one-time signature
- Verification
  - New transactions must verify existing transactions in order ot be confirmed (essentially just verifying that new transaction aren't trying to cheat)
  - To select the transactions to verify, IOTA uses the Markov Chain Monte Carlo (MVCMC) algorithm to find existing transactions to verify that have not been verified yet
  - MCMC designed to discourage/ignore lazy transactions that try verifying heavily verified transactions (to improve their chance of being confirmed (i.e. close to or at the genesis node)), while also not ignoring valid transactions by being too picky
- Consensus
  - Proof of work: certain size hash must be found to validate a transaction, intended to prevent spam attacks

3.5 HashGraph

- HashGraph boasts itself as an alternative way to achieve a distributed ledger
- HashGraph becomes a 'graph' by not excluding forks (as a Blockchain would)
- Using the Gossip protocol, the HashGraph 'weaves' branches back into the main chain
- HashGraph doesn't use proof-of-work or proof-of-stake. Virtual voting is used to reach consensus
- Virtual Voting:

  3 steps:
  - Divide rounds
  - Decide fame
  - Find order

  Divide rounds:
  - Two concepts, rounds and witnesses
  - The first event for a member's node is that node's first witness
  - The first witness is the beginning of the first round for that node
  - All subsequent rounds are part of that first round until a new witness is discovered
  - A witness is discovered when a node creates an event that can strongly see $\frac{2}{3}$ of the witnesses in the current round

  Decide fame:
  - A witness must be either a famous witness, or not
  - If many of the witnesses in the next round can see a witness, it has a high chance of being famous
  - Votes are cast on the fame of witnesses ($\frac{2}{3}$ of votes need to be cast to determine a witness' fame or infamy)

  Find order:
  - Events are sorted into either before the famous witness was decided, or after the famous witness was decided
- Absolute unique order to all transactions

3.6 Hyperledger

- Also known as the Hyperledger project, an initiative by the Linux Foundation to support blockchain-based distributed ledgers
- Numerous project started by the Hyperledger group. Active ones include:
  - Hyperledger Fabric
  - Hyperledger Sawtooth

  - Hyperledger Iroha
- Basically, all about building Blockchains, with various differences in terms of use cases and design choices

3.7 PeerCoin

- Implements both Proof-of-Stake and Proof-of-Work
- Proof-of-Stake was implemented to avoid the 51% attack possibility that exists with Proof-of-Work. With proof-of-stake coin blocks
- Proof-of-Work is still used for the initial minting function, but after 30 days you become eligible to receive blocks from the proof-of-stake implementation
- I don't think anything stops you from mining after this 30 days, so you can still be pumping out the proof-of-work blocks while also 'double dipping' and getting block rewards from proof-of-stake
- The more people mining, the smaller the block reward is
- Currently, the PeerCoin blockchain does use a considerable amount of energy, but the idea is for it to get more energy efficient over time, as miners become "stakeholders" and reduce their proof-of-work SHA solving since they're getting rewards from proof-of-stake anyway

3.8 Ripple

- Ripple is a real-time gross settlement system and currency exchange network
- Ripple existed before blockchains, and its main goals are to provide speed, scalability, transparency and stability. With an emphasis on the first two
- Ripple required two parties for a transaction to occur
  - A regulated financial institution holds funds and issues balances on behalf of customers
  - Second, market makers, which can be hedge funds or currency traders that provide liquidity in the currency they want to trade in
- In Ripple, users make payments in transactions denominated in either fiat transactions or Ripple's internal currency, XRP
- When transactions are made in XRP, Ripple can account for the whole transaction history in its internal ledger
- For other currencies, Ripple can account for the whole transaction history in its internal ledger
- For other currencies, Ripple can only amount for amounts owed (not balances of the users)
- Components of the Ripple Network:

  Server: Like a node, runs the Ripple server. Can act as a validator or a spectator

  Unique Node List (UNL): A list of servers that a server communicates with to form consensus. These servers have to be "honest"

  Proposer: A server that broadcasts transactions. Every server tries to include every valid transaction. Only proposals from servers on the UNL are considered

  Ledger: A record of the amount of currency in each user's account. Each ledger contains:
  - A set of transactions
  - Account Information
  - Timestamp
  - Ledger Number
  - A status bit (to denote whether the ledger is validated or not)

  Consensus and Validating Servers: Responsible for determining consensus. Verifying proposed changes. Needs to meet escalating consensus
- Ripple has a protection to avoid an attacker spamming XRP transactions, for each XRP transaction, a commission is charged, so attackers will eventually run out of XRP
- Biggest blow to Ripple is that while it's distributed, it's not at all decentralized. Ripple owns and runs most of the nodes and maintains a significant percentage of XRP

3.9 Quorum

- An enterprise focused version of Ethereum. A permissioned blockchain of known participants
- Developed to provide the Financial Services Industry with a permissioned implementation of Ethereum that supports transaction and contract privacy
- The primary features of Quorum, and therefore extensions over public Ethereum, are:
  - Transaction and contract privacy

  - Multiple voting-based consensus mechanisms
  - Network/Peer permissions management
  - Higher performance
- Quorum currently includes the following components:
  - Quorum Node (modified Geth Client)
  - Constellation – Transaction Manager
  - Constellation – Enclave
- Constellation is a message transferring agent that implements PGP encryption to allow parties to send private transactions between approved parties while storing a hash of the transaction on the public blockchain (integrity) and private transaction data locally (off chain)
- Enclave acts as a virtual hardware security module, and manages all the encryption/decryption in an isolated manner, working hand-in-hand with the transaction manager

3.10 QTUM

- A Bitcoin core fork, this implements the Proof-of-Stake model for consensus
- Attempts to take the good parts of Ethereum, and the good parts of Bitcoin
- Boasts the ability to execute smart contracts on mobile devices, claiming that they are the only truly decentralized Blockchain
- Allows for the development of distributed apps, which are applications in the QTUM blockchain implemented via smart contracts

3.11 Password cracking using probabilistic context-free grammars

- The paper defines a good way to generate passwords based on previously disclosed password databases
- Context free grammars are useful for generating a probabilistic way to determine what may follow sets of strings
- This paper wasn't actually covered, just suggested

3.12 Spectre and Meltdown Attacks

3.12.1 Spectre

- Attack on other programs or same programs memory using the Speculative Executor and CPU Cache
- Speculative Executor sees a branch and executes the code inside the branch regardless of the output of the branch condition. Result is stored on CPU Cache
- Once the code executes the branch condition, the data is removed from CPU Cache
- Scan the CPU Cache and check for values using reading time, if read time is low then the value is in the cache (and a the value of speculative execution)
- Speculative Execution can inject on a branch forcing it to speculative execute and read the result of that branch
- Can run in V8 engine (Javascript), KVM (Kernel Virtual Machine)
- Target program needs to run on same core (can flood the CPU with programs running 100%)

3.12.2 Meltdown

- More focused at using the Out-of-order execution and CPU Cache
- Out-of-order runs lines of code at the same time or before other lines which might take time
- Allows functions to be run that grant higher level permissions without having those permissions
- Requires no to minimal knowledge of the computer system

3.12.3 Differences

- Spectre requires knowledge of the target program or host and how memory is handled and stored, Meltdown not as much
- Spectre is difficult to patch for and has significant speed impacts
- Spectre is used for accessing memory, Meltdown can access permissioned content
- Meltdown is more targeted to kernel space

3.13 Signal Protocol

- Describes the Signal protocol used for end-to-end encryption in WhatsApp, Messenger and Google's Allo
- Combines the Double Ratchet Algorithm, prekeys and a triple Diffie-Hellman (3-DH) handshake
- Supports end-to-end encrypted group chats as well, using pairwise double ratchet and multi-cast encryption
- Key idea to ratcheting is that the session keys are updated with every message sent. It's called a double ratchet because each participant of the conversation can update the ratchet by sending a message

  "This effectively forces the attacker to intercept all communication between the honest parties, since [they] loses access as soon as one uncompromised message is passed between them...Future Secrecy, or Post-Compromise Security" -Wikipedia
- Public Key Types:

  Identity Key Pair: Generated at install time

  Signed Pre-Key: Generated at install time, signed by the identity key and rotated on a periodic timed basis

  One-Time Pre-Keys: A queue of key pairs for one time use, generated at installed time and replenished as needed

  Root Key: Used to create chain keys

  Chain Key: Used to create message keys

  Message Key: 80 byte key used to encrypt message contents (32 bytes for an AES-256 key, 32 bytes for a HMAC-SHA256 key, 16 bytes for an IV (Initialization Vector))
- Establishing a session:
  1. The initiator requests the public identity key, public signed pre key and a single public one-time pre key for the recipient
  2. The server returns the values. A one-time pre key is only used once, so it is removed from the server after it is requested. If the recipients latest batch of one-time pre-keys has been consumed no one-time pre-key can be returned
  3. The initiator saves the recipient's identity key as `Irecipient`, the signed pre-key as `Srecipient`, and the one-time pre-key as `Orecipient`
  4. The initiator loads its own identity key as `Iinitiator`
  5. The initiator calculates a master secret as: `master_secret = ECDH(Iinitiator, Srecipient) || ECDH(Einitiator, Irecipient) || ECDH(Einitiator, Srecipient) || ECDH(Einitiator, Orecipient)` Where `ECDH` stands for Elliptic-curve Diffie-Hellman. If there is no one-time pre-key the last operation is omitted

Chapter 4

Paper Summaries

4.1 #20 On Scaling Decentralized Blockchains

- Tackles the question of whether blockchains can be scaled up to match the performance of a mainstream payment processor, and what it takes to get there
- Finds that fundamental protocol redesign is needed for blockchains to scale significantly while retaining their decentralization
- Identifies a three-way trade-off among consensus speed, bandwidth, and security. You can do two well, but usually one is traded off
- Separates the different areas of a Blockchain system into 5 distinct planes: Network, Consensus, Storage, View and Side (Planes)
- **Network Plane:** role of propagating transaction messages, specifically valid transactions. Two major inefficiencies:
  - To avoid DoS attacks, where an invalid transaction is attempted to be propagated, a node must fully receive and attempt to validate the transaction before ignoring it if it's invalid
  - Transactions are propagated, and then later, a block is propagated when it is mined (which contains all the previously propagated transactions). Each transaction is transmitted twice
- **Consensus Plane:** the role of mining blocks and verifying their legitimacy and addition to the Blockchain
  - Held back by proof-of-work, which facilitates the 'three-way trade-off' identified above. Changing this has the potential to overcome this problem
- **Storage Plane:** essentially a 'global memory' that stores and provides availability for authenticated data produced by the Consensus Plane
  - Essentially the distributed ledger
  - Weakness with Bitcoin's distributed ledger is that each node stores the entire ledger, resulting in many duplicates
- **View Plane:** facilitates the function of a view over the UTXO (unspent transaction outputs) set. It has a lot of similarities to the Storage Plane
- **Side Plane:** allows off-the-main-chain consensus

4.2 #21 On Bitcoin Security in the Presence of Broken Crypto Primitives

Abstract

- Digital currencies reply on cryptographic primitives
  - A cryptographic primitive is set of low-level cryptographic algorithms that are used to frequently build cryptographic protocols (hash functions, encryption/decryption functions)
- Cryptographic primitives don't last forever. Increased computational power and advanced cryptanalysis cause these primitives to break
- Important for a crypto currency to anticipate such breakage
- Depending on the primitive and type of breakages, a range of effects are possible. From minor privacy violations to a complete breakdown of the currency

4.3 #22 Bitcoin and The Age of Bespoke Silicon

FPGA: Field Programmable Gate Arrays
ASIC: Application Specific Integrated Circuits

- Progression of mining: CPU $\rightarrow$ GPU $\rightarrow$ FPGA $\rightarrow$ ASIC
- GPUs have limitations:
  - Requires computer components to run (Motherboard, CPU, RAM, ...)
  - Most other boards only have 1 or GPU slots
  - High energy usage in combination with other computer hardware
  - Cooling and hardware failure

- ASICs were initially crowd funded and produced by inexperienced companies
- Takes note that hardware innovation is stagnating because new ideas are expensive to test

4.4 #23 A Fistful of Bitcoins: Characterizing Payments Among Men with No Names

- Almost impossible to get and trade Bitcoins without giving personal information to an entity (entity, pool, bitcoin exchange, etc)
  - This is assuming you don't solo-mine
- Bitcoin blockchain is far from anonymity

4 ways to mask Bitcoin transactions (all methods incur transaction costs)

Splitting: Split one "Dirty Wallet" into multiple smaller "Dirty Wallets"

Folding: Combining multiple "Dirty Wallets" and "Clean Wallets" together

Aggregating: Combining multiple "Dirty Wallets" into one Aggregated Wallet

Peeling: Split one "Dirty Wallet" into smaller wallets, then each smaller wallet do a change address

Transaction cost has changed from when these were first discovered ($0.20 USD $\rightarrow$ $2.30 USD)

4.5 #24 An Analysis of Anonymity in Bitcoin Using P2P Network Traffic

Phase 0: Prune transaction data to remove potential sources of noise

Phase 1: Using relay patterns we have observed for transactions, hypothesize an "owner" IP for each transaction

Phase 2: Break transactions down into their individual Bitcoin addresses. We do this to create more granular (Bitcoin address, IP) pairings

Phase 3: Compute statistical metrics for our (Bitcoin address, IP) pairings

Phase 4: Identify pairings that may represent ownership relationships

Phase 5: Eliminate ownership pairings that fall below our defined thresholds

- Creates a Bitcoin client, CoinSeer, to collect data of the peers in the network (collects about 60GB of data per week)
- Able to map between 252 to 1162 Bitcoin addresses to IP addresses that likely owned these addresses
- Using ip masking services or online wallets reduces the accuracy of this
- No mention of dynamic IPs

4.6 #25 Mixcoin

- Protocol to make anonymous payments easier in Bitcoin and similar cryptocurrencies
- Builds on currency mixing
- Adds a mechanism for exposing theft in coins
- Provides anonymity between mixing coins against a passive attack
- Similar anonymity to traditional communication mixes against an active attack

Active attack: The attackers tries to modify the system under threat, either by increasing priviledges or changing information

Passive attack: The attacker listens and gathers information from the system without actually modifying the system

4.7 #26 A survey of attacks on Ethereum smart contracts

Vulnerabilities in Solidity

- **Call to the Unknown:** Some of the primitives used in Solidity to invoke functions and to transfer ether may have the side effect of invoking the fallback function of the callee/recipient
- **Gasless Send**
- **Exception Disorders**
  - In Solidity there are several situations where an exception may be raised: (the execution runs out of gas, the call stack reaches its limit, the command throw is executed)
  - However, Solidity is not uniform in the way it handles exceptions: there are two different behaviors, which depend on how contracts call each other
- **Type Casts:** The Solidity compiler detects some type errors, but not others. Even in presence of type errors, the EVM doesn't throw error at runtime

- **Re-entrancy:** The atomicity and sequentiality of transactions may induce programmers to belive that, when a non-recursive function is invoked, it cannot be re-entered before its termination. However, this is not always the case, because the fallback mechanism may allow an attacker to re-enter the caller function
- **Keeping Secret:** Fields in contracts can be public, i.e. directly readable by everyone, or private, i.e. not directly readable by other users/contracts. Still, declaring a field as private does not guarantee its secrecy

Vulnerabilities in EVM

- **Immutable bugs:** Once a contract is published on the blockchain, it can no longer be altered
- **Ether lost in transfer:** When sending ether, one has to specify the recipient address, which takes the form of a sequence of 160 bits. However, many of these addresses are orphans, i.e. they are not associated to any user or contract. If some ether is sent to an orphan address, it is lost forever (note that there is no way to detect whether an address is orphan).
- **Stack size limit:** Each time a contract invokes another contract the call stack associated with the transaction grows by one frame. The call stack is bounded to 1024 frames: when this limit is reached, a further invocation throws an exception

Vulnerabilities in Blockchain

- **Unpredictable state:** The state of a contract is determined by the value of its fields and balance. In general, when a user sends a transaction to the network in order to invoke some contract, he cannot be sure that the transaction will be run in the same state the contract was at the time of sending that transaction
- **Generating randomness:** EVM bytecode is deterministic, so for generating of random numbers an initialization seed is chosen uniquely for all miners, based on the details of the block. A malicious miner/group of miner could create a block with the intention of biasing the outcome of this random seed
- **Time constraints:** Miners can choose the timestamp for the block mined, which can cause issues with various smart contracts, particularly when functions are dependent on specific times

Common cause of insecurity in smart contracts is the difficulty in detecting mismatches between intended and actual behavior, a non-Turing complete, human readable language could help overcome this issue.

4.8 #30 Cold Boot Attacks on Encryption Keys

Halderman et al., Lest We Remember: Cold Boot Attacks on Encryption Keys, USENIX Security Symposium, 2008

Abstract

"Contrary to popular assumption, DRAMs used in most modern computers retain their contents for several seconds after power is lost, even at room temperature and even if removed from a motherboard. Although DRAMs become less reliable when they are not refreshed, they are not immediately erased, and their contents persists sufficiently for malicious (or forensic) acquisition of usable full-system memory images.

We show that this phenomenon limits the ability of an operating system to protect cryptographic key material from an attacker with physical access. We use cold reboots to mount successful attacks on popular disk encryption systems using no special devices or materials. We experimentally characterize the extent and predictability of memory remanence and report that remanence times can be increased dramatically with simple cooling techniques.

We offer new algorithms for finding cryptographic keys in memory images and for correcting errors caused by bit decay. Though we discuss several strategies for partially mitigating these risks, we know of no simple remedy that would eliminate them."

Notes

- DRAMs typically lose their contents over a period of several seconds, if the chips are cooled to low temperatures (-50°C), the data can persist for minutes - hours.

- The researchers used non-destructive disk forensics techniques to create memory images, and extract the keys needed to decrypt several popular disk encryption systems (BitLocker, TrueCrypt and FileVault)
- If a system is rebooted, often the BIOS will overwrite portions of memory, and some systems are configured to perform a destructive memory test during its Power-On Self Test (POST)
- A warm boot is initiated by the host operating system and gives the OS a chance to cleanly exit application and wipe memory. A cold boot is initiated either by pressing the system restart button, or temporarily removing the power, this gives the OS not opportunity to scrub memory state.
- In order to create images of the memory, the DRAM in a running system is first cooled, then a cold boot is initiated and the system is booted into a special forensics tool via PXE network boot/USB etc. The cooled DRAM could also be removed an installed into another machine, bypassing any BIOS/POST protections.
- Algorithms were developed to extract cryptographic keys and correct bit errors in the range of 5% - 50%, this is achievable by comparing the key to other key precompute schedules stored in memory. ie. RSA's p + q values are often stored alongside the private key in order to perform faster computations.
- The paper describes in detail the process for reconstructing DES, AES, RSA and tweak keys.
- A method was developed in order to identify keys in memory, even in the presence of bit errors. This is done by again looking for additional key schedules, searching for blocks of memory that closely satisfy the combinatorial properties of a valid key schedule
- The paper also describes in detail the process for identifying AES, DES, RSA and file system encryption keys in a memory dump.
- Countermeasures discussed include:

  Scrubbing Memory: software should overwrite keys when they are no longer needed, and prevent keys from being paged to disk

  Limiting booting from external media: the majority of attacks were only possibly by booting into the forensics tools

  Suspending a system safely: require a password in order to wake a suspended computer, memory should be encrypted during suspension with a key derived from the password.

  Avoid precomputation: precomputations should be cached for a certain period and scrubbed if not re used.

  Key Expansion: Apply some transform to keys when they are stored in memory in order to make it more difficult to reconstruct

  Physical Defence: Lock/Encase the DRAM to prevent removal

  Architectural changes: Design DRAM that loses its state very rapidly past the intended refresh interval

  Encrypting in the disk controller: Perform disk encryption operations in disk controller rather than by software in the main CPU, and storing the keys in the disk controller rather than DRAM.

  Trusted computing: Deploying trusted computing hardware will help the machine determine whether it is safe to store keys in DRAM at this time or not.

4.9 #31 Vanish: Increasing Data Privacy with Self-Destructing Data

- Data is cached and stored on third party machines
- Developed a program called Vanish
- Vanish encrypts messages after a particular amount of time
- If an attacker obtains a cached copy of the message, the user's cryptographic keys, and passwords; the message can't be decrypted
- Both a Firefox plugin and File application proof-of-concepts were made

4.10 #32 Android Security: A Survey of Issues, Malware Penetration and Defences

- Security features provided by Android:
  - Application sandboxing
  - Permissions at framework-level
  - Secure system partition
  - Secure Google Play Store

  - Various other security enhancements over the years:
    * Mandatory Access Control (MAC) policies since 4.3 (Jelly Bean)
    * Authentication required when using Android Debug Bridge
    * Removing `setuid()` and `setgid()` functions
- Security issues faced by the Android platform
  - Updates
    * Original Equipment Manufacturers (OEMs) have the responsibility to provide updates to the consumers who provide their product
    * Even though Android itself is open source, and freely distributed, many of these OEMs add further modifications to suit their business interests
    * It may even suit an OEM to not release an update, if there is no financial reason to do so
    * This leads to fragmentation, where some devices are able to update, and some aren't, and the proliferation in exploits and vulnerabilities as it becomes worthwhile to attack older Android OS's
  - Native Code Execution: In older Android OS's, native code execution can execute publicly at the root level
  - Types of Threats
    * Privilege Escalation attacks
    * Privacy leaks through permissions
    * Malicious apps can spy on users
    * Malicious apps can use the device to make phone calls/send messages
    * Colluding attacks
    * Denial of service attack
- Malware Penetration Technique
  - Repackaging popular apps
  - Drive-by download
- Various ways to detect, assess and analyse Android applications, the best ones usually are off-device
- Two main methods of detection:

  Static: Utilizes control-flow and data-flow analysis to detect improper patterns in an application's design. Can be less successful if the app is encrypted or uses transformation techniques

  Dynamic: Executes applications in a sandboxed environment in order to monitor activities and identify anomalous behavior. Can be less successful if the app uses anti-emulation techniques

4.11 #33 Computing Arbitrary Functions of Encrypted Data

- Allows hosts to run functions on sets of encrypted data without decrypting the data first
- Functions need to written to handle the encrypted data
- Hosts don't need to encryption details
- Makes cloud computing more compatible with privacy
- Very inefficient (Addition/Subtraction takes seconds, multiplication takes minutes, division takes an hour)

4.12 #34 The EigenTrust Algorithm for Reputation Management in P2P Networks

- Assigns a trust value to a P2P uploader
- Network can effectively identify malicious peers and isolate them from the network
- Based on Power iteration
- Simulations has shown to be effectively, even when malicious peers cooperate in an attempt to deliberately subvert the system

4.13 #35 Detecting and Defending against third-party tracking on the web

- Estimates that trackers capture up to 20% of a user's browsing behavior
- Two main types of trackers: within-site trackers (like Google Analytics), and cross-site trackers (like DoubleClick)
- Firefox addon, ShareMeNot, which drastically mitigates the prevalence of third-party social widget tracking

Category of trackers:

Analytics: Within-Site, Serves as third-party analytics engine for sites

Vanilla: Cross-Site, Uses third-party storage to track users across sites

Forced: Cross-Site, Forces user to visit directly (e.g. via popup or redirect)

Referred: Cross-Site, Relies on a **Vanilla**, **Forced**, or **Personal** tracker to leak unique identifiers

Personal: Cross-Site, Visited directly by the user in other contexts

4.14 #36 Chip and PIN is broken

- EMV is the dominant protocol used for smart card payments worldwide
- Secures transactions by authenticating card and customer using a combination of cryptographic authentication codes, digital signatures, and entry of a PIN
- A man-in-the-middle attack is possible, trick the terminal that the PIN was entered while telling the card it wasn't entered
- The attack is possible because the PIN is never explicitly authenticated
- The attack works by intercepting communication and modify bits that say the card is authenticated
- Many banks deny this attack is possible saying the card cannot be used without the correct PIN

3 phases of the EMV protocol:

Card authentication: Tells the terminal which bank to communicate with, and that the card is valid

Cardholder verification: Tells the terminal that the PIN entered (into the terminal) is the correct PIN for this card

Transaction authorization: Tells the terminal that the bank that issues this card will subsequently authorize this transaction

Key steps of the attack:

1. Card reader is required to read the data of a legit card
2. That data is fed into a (python) program which pipes the information to an FPGA
3. FPGA programs the data onto a dummy card
4. The dummy card allows the attack to listen to the communications between card and terminal
5. When the (python) program sees the terminal send the PIN, the attack begins $\rightarrow$ the card itself sees that no pin was entered

4.15 #37 Experimental Security Analysis of a Modern Automobile

- Many Electronic Control Units (ECUs) communicate together over an interval vehicular network
- Hacking one of these devices allows the potential to affect other devices
- Removal or modification of critical-safety features (stopping of individual wheels, stopping the engine, disabling brakes)
- Able to embed code into the car that will erase itself after a crash occurs
- Attacks can bypass network security features and bridge between interval vehicular subnets

4.16 #38 Side-Channel Leaks in Web Applications: a Reality Today, a Challenge Tomorrow

- Software-as-a-service is becoming mainstream and more applications are delivered through the Web
- A web application contains browser-side and server-side components
- Part of the application's internal information flows are exposed on the network
- Despite encryption, side-channel attacks can leak user information
- Types of information already being leaked out: Healthcard, taxation, investment, and web searches
- From this illnesses/medications/surgeries/family income/investments despite HTTPS encryption and WPA/WPA2 WiFi encryption
- Root causes are fundamental characteristics of web applications: stateful communication, low entropy, and significant traffic distinctions.
- Works by checking which information is repeated from other users and generating an ambiguity set (e.g. Male/Female would generate two sets of data which be relatively split 50/50)
- Autocomplete helps side-channel attacks because the attack relies on changes to the users' state
- Most solutions for this attack are application-specific, however padding can be added to

information (but this increases overhead and bandwidth usage)

4.17 #39 A convenient method for securely managing passwords

- Provides a password manager program
- Users enter a master password and then get the individual password for that site
- Protects against leaked passwords from different sites
- Generated passwords are more protected against dictionary and brute force attacks
- Instead of the user providing a password for each site (like normal password managers), this generates a password based on the site you are at
- Combines site name, username and the master password with a hash function to generate the password
- Minimal support for periodic password changes
- Changing master password requires changing all the passwords
- $k1$ is used during initialization, it is the number of times username and master password are concatenated (cached for session of user)
- $k2$ is used during password generation, number of times site name, concatenated with master password, concatenated with with result

4.18 #40 The TESLA Broadcast Authentication Protocol

- **TESLA:** Times Efficient Stream Loss-tolerant Authentication
- Broadcast protocol that allows for source authentication
- Low communication and computational overhead
- Easily scaled to large number of receivers, and tolerates packet loss
- Uses symmetric cryptographic functions (MAC functions)
- Assume all network nodes are loosely time synchronized
- Time synchronization is required because there needs to be a delay in transmitting the key, if the attacker knows when the key is sent otherwise they could pretend to be the attacker
- An attacker could flood the sender with time synchronization requests, leading to a DoS attack
- Buffering of packets is needed because the key isn't sent till later

Order of execution:

1. An authentication code is appended to the packet (only sender knows)
2. Receiver will receive a packet and not know how to authenticate
3. Later sender will send the key to the receiver, allowing for packet authentication

4.19 #41 Anonymous Connections and Onion Routing

- Provides anonymous connections that are resistant to both eavesdropping and traffic analysis
- Onion routing is bidirectional, near real-time, used anywhere a socket connection can be used
- Onion refers to the data structure
- Routers used for onion routing are called onion routers
- An Onion appears different to each onion router
- Proxy-aware applications (web browsers, email clients) require no modification to use onion routing

  They use a set of proxies to onion route

4.20 #42 Honeywords: Making Password-Cracking Detectable

- Creates a set of passwords for each account which are "honeyword" passwords
- When an attacker gets access to the hash passwords, do not know which is real or honeyword
- If the honeyword is attempted to be used as a login, alarm is set off
- Secondary server is used to check if password is real or honeyword (honeychecker)
- Works only if database is compromised, rather than physical system

- Works for every account compared to honeypot technique
- Open Problems:
  - How to ensure security of the honeychecker system
  - How to maintain the integrity of the honeycheck if a compromised computer system attacks it
  - What to do after the honeyword is entered? Simply disable the triggered account?
  - How to protect against a man-in-the-middle
  - Cam password models underlying cracking algortihms be easily adapted for use in chaffing-with-a-password-model

4.21 #43 RSA Key Extraction via Low-Bandwidth Acoustic Cryptanalysis

- Able to get the full 4096-bit RSA keys used for encryption

  When using the library GnuPG
- Uses small acoustic noises made by electric components
- Very low bandwidth (20kHz to a few hundred kHz), compared to the GHz-scale clock rates
- Takes roughly an hour for a standard attack
- Acoustic shield can be used, but makes cooling difficult
- Noisy environments can be filtered out in the data processing phase
- Parallel software load doesn't help reduce this attack, shifts leakage frequency to a lower range
- Cipher text randomization, works by generating a random 4096 bit number and perform few calculations with it. Attack can't distinguish between the real and random number (Modulus randomization works and is similar to this)
- Padding the ciphertext with 0s or n-bits, attacker won't know then the ciphertext is beginning

4.22 #44 The Web Never Forgets: Persistent Tracking Mechanisms in the Wild

Three advanced web tracking mechanisms:

- Explores three web tracking mechanisms:

  Canvas Fingerprinting: Canvas fingerprinting uses the HTML5 canvas element to draw an invisible image. The site can then call the Canvas' APIs '`ToDataURL`' method to get the canvas pixel data in URL form. Now the site can hash this pixel data, and as long as they unique images (which are invisible to the end user), these can serve as fingerprints for the user accessing the site

  Evercookies: Cookies that actively circumvent a users' attempts to clear cookies by abusing various browser storage mechanisms

  Cookie Syncing: Allows trackers to cooperate with each other, when they see a friendly cookie that doesn't necessarily belong to them. This allows for back-end server-side data merges that are completely hidden from the end user
- Mitigation for:

  Canvas Fingerprinting: The Tor browser simply notifies a user that a website attempted to draw using the HTML5 canvas tool, and allows the user to accept/decline the attempt (since this tool also has legitimate uses)

  Evercookies: The straightforward way is to just clear all browser storage locations. Depending on which browser you use, this may be hard to achieve. If you use Adobe Flash, this is not a robust solution as the storage Flash uses can be utilized by multiple browsers and is therefore not isolated to one

  Cooking Syncing: No robust way to stop cookies from cooperating. EFF's Privacy Badger add-on uses a heuristic method to block third-party cookies. Another way is to just not allow any sort of third-party traffic to store on your computer, but this can have negative impacts on certain websites, and will be frivolous if you already have certain cookies already on your system

4.23 #45 Sound-Proof: Usable Two-Factor Authentication Based on Ambient Sound

- Two-factor accounts protects accounts even if passwords are leaked
- Most people don't like the extra step required to login
- Either people have to get a code from their phone or install a program on their computer
- SoundProof doesn't require the user to interact with their phone

  Second authentication is the proximity of phone to device logging in
- Uses and compares ambient noise to check proximity
- Survey favored Sound-Proof over Google 2-Step and majority would be willing to use Sound-Proof even for scenarios which two-factor is optional

4.24 #46 Rocking Drones with Intentional Sound Noise on Gyroscopic Sensors

- Target drones gyroscope at a frequency under 30kHz
- One out of two drones were affected under testing (drones were using targeted gyroscopes, ran 20 trials)
- Attack distance of 37.58m if 140dB of SPL (Sound Pressure Level) at 1m is used
- Ways to prevent the attack:

  Physical Isolation: Reduce the amount of noise that can reach the device, 1 inch thick foam

  Different Sensor: Use a different sensor comparator that only responds to resonant frequency

  Resonance Tuning: Change the resonance frequency of the gyro by adding a capacitor

4.25 #47 IoT Goes Nuclear: Creating a ZigBee Chain Reaction

- Able to infect device have that device spread the virus across the network
- The devices automatically share the update between each other as an update
- Once infected, the attacker is capable of turning all the lights on or off, permanently brick them, or exploit them in a DDoS attack
- There is a required number of devices (15,000) in close proximity (105 square kilometers)
- Tested and working with Phillips Hue Bulbs

4.26 #48 Game of Drones - Detecting Streamed POI from Encrypted FPV Channel

- Analyzes the bitrate of a wireless transmission between drone and controller
- Using physical stimuli (e.g. flash a window), detect the changes in bitrate
- If the bitrate increases then the drone is aimed at the physical stimuli

4.27 #49 Understanding the Mirai Botnet

- Composed primarily of embedded and IoT devices
- Massive DDoS attack in late 2016
- Expected peak infection of 600k devices
- Proves that novice malicious techniques can compromise enough low-end devices
- Mirai was successful because of how underdeveloped security was for the beginning of IoT devices
- Seen as a call to arms to increase security standards of IoT devices

4.28 #50 Client Puzzles: A Cryptographic countermeasure against connection depletion attacks

- Provides a solution to TCP SYN flooding
  This is DoS attack where the attacker opens a series of connections and never closes them
- The solution is to send a cryptographic puzzle to the client, the client must solve this puzzle before the connection is initiated
- The time for the client to solve the puzzle is greater than the server to generate a new puzzle

Chapter 5

Do you need a Blockchain?

- Essentially Blockchain is only suitable for any system that requires a database or someway to store data, multiple writers (having only one writer would better suit using a regular database as better throughput) and no Trusted Third Party (TTP) that can always be relied on for writing. Otherwise don't use blockchain
- Types of Blockchain (Private/Public $\rightarrow$ Reader, Permissioned/Permissionless $\rightarrow$ Centralized)
  - Permissionless Blockchain
  - Public Permissioned Blockchain
  - Private Blockchain

5.1 What are some examples where a blockchain would be necessary?

DOAs (Decentralized Autonomous Organizations) CAN be useful, although they likely wouldn't need to use a full fledged permissionless blockchain in most cases

Finance (interbank payments): Multiple parties that don't trust each other that need high number of readers but not necessarily high throughput. Although this would certainly still use the central bank only has a TTP to authorize other banks and be a permissioned blockchain, public vs private is a matter of public opinion

IoT: Payment systems between machines, smart cars charging at electric charging stations, small devices requiring data processing from untrusted server. However, sensors play a huge role and thus need to be trusted by all parties, thus this depends on a case by case basis

Voting: Multiple parties not trusting the outcome as valid, need a mechanism that is able to verify votes, where they came from, only one vote spent, cannot forge votes or create new ones. Essentially system requires public verifiability as many mutually untrusted parties exist

Smart Contracts: Well suited as no need for TTP. Either permissioned or permissionless

Multi-party Trade: Exchanging digital goods without trusted dispute mediator, physical goods still require TTP to handle disputes

Proof of ownership: In the case of patenting, it would be handy to have a verifiable record, however this does not always fully prove ownership

5.2 Properties of Distributed Ledgers vs Centralized Systems

- Public verifiability
- Transparency
- Privacy
- Integrity
- Redundancy
- Trust Anchor

Chapter 6

Past Exams

6.1 2017 Sample

6.1.1 Question 1

Alice, Bob, Charlie and Debbie use Shamir's method to implement a (2, 4)-threshold secret sharing scheme to share the secret K. Computations are done in GF(7), i.e. modulo 7. Bob's and Charlie's shares are as follows:
Bob: $(x_B, y_B) = (3, 5)$
Charlie: $(x_C, y_C) = (5, 4)$
Compute the shared secret K

$$
\begin{aligned}
5 &= 3m + c && \text{(Bob)} \\
\frac{5-c}{3} &= m \\
\frac{5}{3} - \frac{1c}{3} &= m \\
\frac{12}{3} - \frac{15c}{3} &= m \\
4 - 5c &= m \\
m &= 4 - 5c \\
4 &= 5m + c && \text{(Charlie)} \\
4 &= 5(4 - 5c) + c && \text{(Sub Bob into Charlie)} \\
4 &= 20 - 25c + c \\
4 &= 20 - 24c \\
4 &= 6 - 3c \\
-2 &= -3c \\
2 &= 3c \\
\frac{2}{3} &= c \\
\frac{30}{3} &= c \\
10 &= c \\
3 &= c
\end{aligned}
$$

6.1.2 Question 3

Blind Signatures are used to enable a user to verify the a set of information without knowing what that information is. This is used to maintain anonymity in cryptocurrencies where a coin will be blindly signed by the bank to ensure they can't trace payments. The RSA signature process uses a blind signature to get information signed without the singer knowing what they are signing.

6.1.3 Question 4

In a Shamir secret sharing scheme, the secret is the constant term c of a polynomial of degree 4. Computations are done in GF(991), i.e. modulo the prime 991. Suppose Alice, Bob and Susie have the following three shares: $(2, 197), (4, 874), (13, 547)$. How many possibilities are there for the secret? How much information can be gained from the three shares? Explain your answer.

Because the polynomial is of degree 4, there is still one point of freedom which does not provide any additional knowledge of the curve. The only known information is that the secret lies somewhere between 0 and 991.

6.1.4 Question 5

Generate two valid shares for a secret $K = 30$ in a (3, 4)-threshold secret sharing scheme (Blakely's method). Computations are done in GF(37). Explain the individual steps involved.

$$y = m_1x^3 + m_2x^2 + m_3x + c$$

Share 1: $y = 5x^3 + 3x^2 + 10x + 30$
Random points: $(1, 48), (2, 102), (10, 5430) \rightarrow (1, 11), (2, 28), (10, 28)$