

# Contents

<b>1 Exam Structure</b>	<b>3</b>
1.0.1 What is Covered? . . . . .	3
1.0.2 Format . . . . .	3
<b>2 Lecture Notes</b>	<b>4</b>
2.1 Lecture 1: Bitcoin, Cryptocurrencies Blockchain Technology . . . . .	4
2.1.1 Digital Cash . . . . .	4
2.1.2 eCash . . . . .	4
2.2 Cryptographic (one-way) Hash Function . . . . .	5
2.2.1 Ideal One-way Cryptographic Hash Function “Random Oracle” Model . . . . .	5
<b>3 Paper Summaries</b>	<b>6</b>
3.1 #20 On Scaling Decentralized Blockchains . . . . .	6
3.2 #21 On Bitcoin Security in the Presence of Broken Crypto Primitives . . . . .	6
3.3 #22 Bitcoin and The Age of Bespoke Silicon . . . . .	6
3.4 #23 A Fistful of Bitcoins: Characterizing Payments Among Men with No Names . . . . .	7
3.5 #24 An Analysis of Anonymity in Bitcoin Using P2P Network Traffic . . . . .	7
3.6 #25 Mixcoin . . . . .	7
3.7 #26 A survey of attacks on Ethereum smart contracts . . . . .	7
3.8 #30 Cold Boot Attacks on Encryption Keys . . . . .	8
3.9 #31 Vanish: Increasing Data Privacy with Self-Destructing Data . . . . .	9
3.10 #32 Android Security: A Survey of Issues, Malware Penetration and Defences . . . . .	9
3.11 #33 Computing Arbitrary Functions of Encrypted Data . . . . .	10
3.12 #34 The EigenTrust Algorithm for Reputation Management in P2P Networks . . . . .	10
3.13 #35 Detecting and Defending against third-party tracking on the web . . . . .	10
3.14 #36 Chip and PIN is broken . . . . .	11
3.15 #37 Experimental Security Analysis of a Modern Automobile . . . . .	11
3.16 #38 Side-Channel Leaks in Web Applications: a Reality Today, a Challenge Tomorrow . . . . .	11
3.17 #39 A convenient method for securely managing passwords . . . . .	12
3.18 #40 The TESLA Broadcast Authentication Protocol . . . . .	12
3.19 #41 Anonymous Connections and Onion Routing . . . . .	12
3.20 #42 Honeywords: Making Password-Cracking Detectable . . . . .	12
3.21 #43 RSA Key Extraction via Low-Bandwidth Acoustic Cryptanalysis . . . . .	13
3.22 #44 The Web Never Forgets: Persistent Tracking Mechanisms in the Wild . . . . .	13
3.23 #45 Sound-Proof: Usable Two-Factor Authentication Based on Ambient Sound . . . . .	14
3.24 #46 Rocking Drones with Intentional Sound Noise on Gyroscopic Sensors . . . . .	14
3.25 #47 IoT Goes Nuclear: Creating a ZigBee Chain Reaction . . . . .	14
3.26 #48 Game of Drones - Detecting Streamed POI from Encrypted FPV Channel . . . . .	14
3.27 #49 Understanding the Mirai Botnet . . . . .	14
3.28 #50 Client Puzzles: A Cryptographic countermeasure against connection depletion attacks . . . . .	15

## Contributors:

- Daniel Fitz (Sanchez)

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

# Chapter 1

## Exam Structure

- 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 Android system and applications
- Describe if and how malicious application can detect the presence of Aurasium

### 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

- Cannot be spent multiple times (“Double Spend” Problem)

### 2.1.2 eCash

# 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, Blockchain 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)

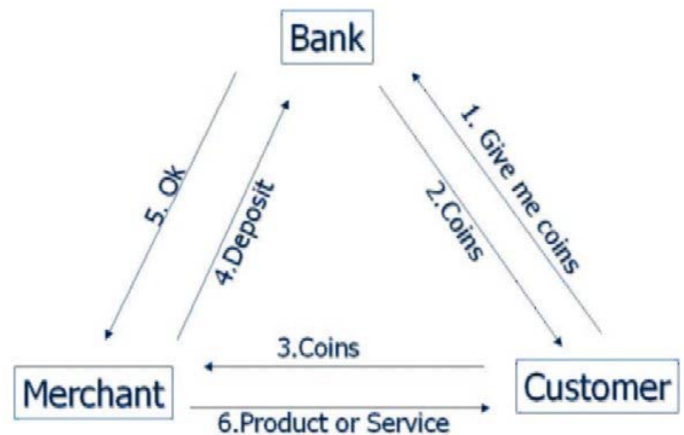


Figure 2.1: eCash Step Through

- **eCash Transaction**  
Each eCash coin has a unique identifier 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

#### 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 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 ‘pre-image 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)

### 2.2.1 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

# Chapter 3

## Paper Summaries

### 3.1 #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 Consen-

sus 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

### 3.2 #21 On Bitcoin Security in the Presence of Broken Crypto Primitives

#### Abstract

- Digital currencies rely 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

### 3.3 #22 Bitcoin and The Age of Bespoke Silicon

**FPGA:** Field Programmable Gate Arrays

**ASIC:** Application Specific Integrated Circuits

- Progression of mining: CPU → GPU → FPGA → 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

### 3.4 #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 → \$2.30 USD)

### 3.5 #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

### 3.6 #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 privileges or changing information

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

### 3.7 #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 believe 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

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.

## 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

## 3.8 #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 and 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.

### 3.9 #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

### 3.10 #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

### 3.11 #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 be written to handle the encrypted data
- Hosts don't need to know encryption details
- Makes cloud computing more compatible with privacy
- Very inefficient (Addition/Subtraction takes seconds, multiplication takes minutes, division takes an hour)

### 3.12 #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 have shown to be effective, even when malicious peers cooperate in an attempt to deliberately subvert the system

### 3.13 #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 add-on, 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

### 3.14 #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 → the card itself sees that no pin was entered

## 3.15 #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

## 3.16 #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)

### 3.17 #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
- $k_1$  is used during initialization, it is the number of times username and master password are concatenated (cached for session of user)
- $k_2$  is used during password generation, number of times site name, concatenated with master password, concatenated with result

### 3.18 #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

### 3.19 #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

### 3.20 #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 honey-pot technique
- Open Problems:
  - How to ensure security of the honey-checker 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
  - Can password models underlying cracking algorithms be easily adapted for use in chaffing-with-a-password-model

### 3.21 #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

### 3.22 #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

### 3.23 #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

### 3.24 #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

### 3.25 #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

### 3.26 #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

### 3.27 #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

### **3.28 #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