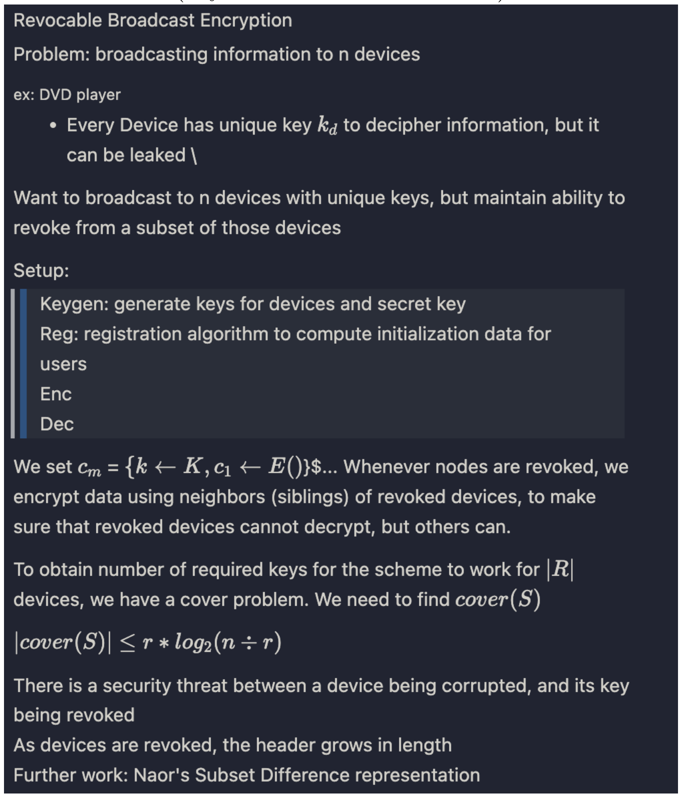
Notes

Harrison Nicholls
Spring 2023

*

Revocable Broadcast encryption (Summary of notes) The goal of revocable broadcast encryption is to revoke a subset of devices while still being able to broadcast to the rest. The system involves generating keys and secret keys, registration algorithms, encryption, and decryption. The number of required keys for the system is obtained using a cover problem. There is a security threat if a device is corrupted, and its key is revoked, and the header grows in length as devices are revoked. Further work includes Naor's Subset Difference representation

(My actual notes from class)



Revocable Broadcast Encryption

Problem: broadcasting information to n devices

ex: DVD player

- Every Device has unique key k_d to decipher information, but it can be leaked \

Want to broadcast to n devices with unique keys, but maintain ability to revoke from a subset of those devices

Setup:

> Keygen: generate keys for devices and secret key
> Reg: registration algorithm to compute initialization data for users
> Enc
> Dec

We set c_m = $\{k \leftarrow K, c_1 \leftarrow E()$}$... Whenever nodes are revoked, we encrypt data using neighbors (siblings) of revoked devices, to make sure that revoked devices cannot decrypt, but others can.

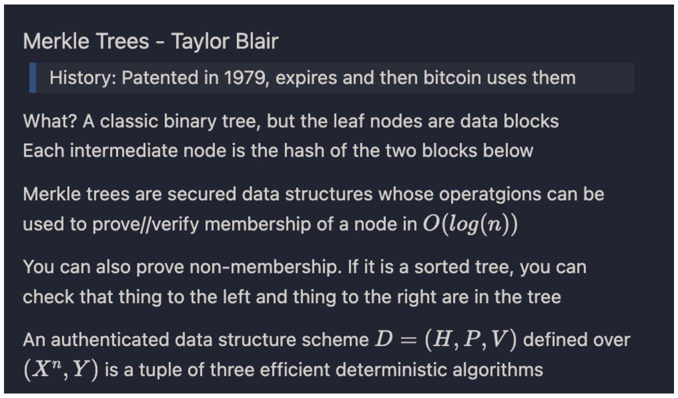
To obtain number of required keys for the scheme to work for $|R|$ devices, we have a cover problem. We need to find $cover(S)$

$|cover(S)| \leq r * log_2(n \div r)$

There is a security threat between a device being corrupted, and its key being revoked
As devices are revoked, the header grows in length
Further work: Naor's Subset Difference representation

Merkle Trees

A Merkle tree is a binary tree structure where the leaf nodes are data blocks and each intermediate node is the hash of the two blocks below. It was patented in 1979 and later used by Bitcoin. Merkle trees provide secure data structures that allow for efficient proof and verification of membership in $O(log(n))$ time, as well as proving non-membership. An authenticated data structure scheme is defined as a tuple of three deterministic algorithms: H, P, and V.



Merkle Trees - Taylor Blair

> History: Patented in 1979, expires and then bitcoin uses them

What? A classic binary tree, but the leaf nodes are data blocks
Each intermediate node is the hash of the two blocks below

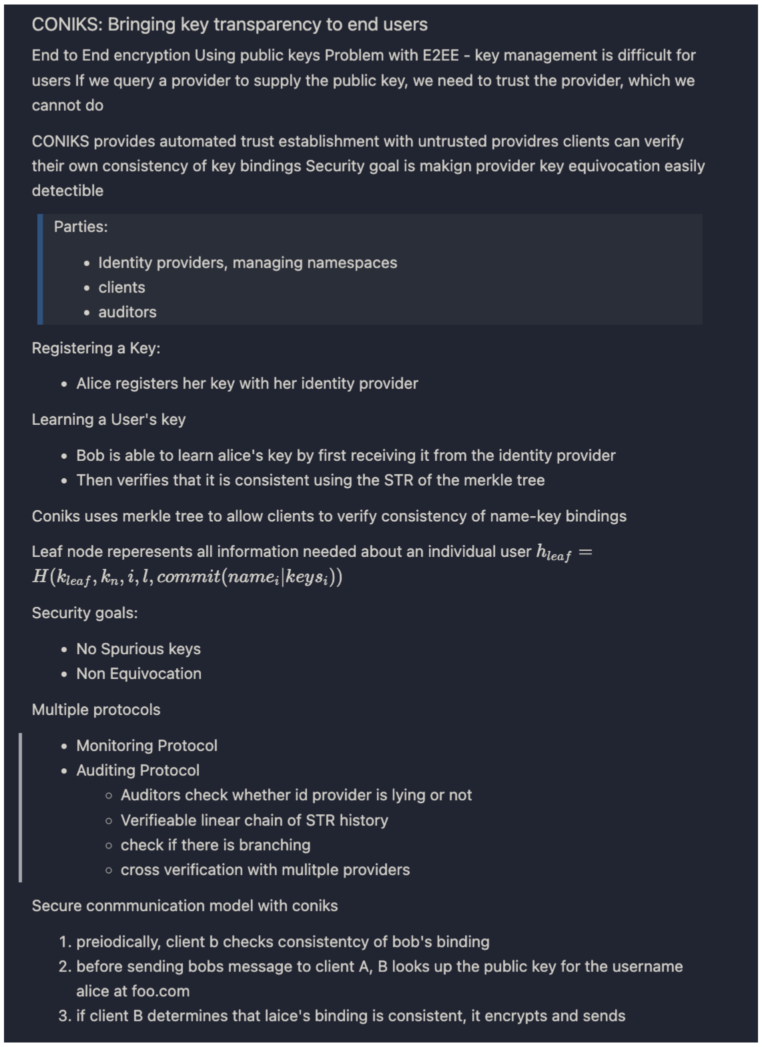
Merkle trees are secured data structures whose operatgions can be used to prove//verify membership of a node in $O(log(n))$

You can also prove non-membership. If it is a sorted tree, you can check that thing to the left and thing to the right are in the tree

An authenticated data structure scheme $D = (H, P, V)$ defined over (X^n, Y) is a tuple of three efficient deterministic algorithms

CONIKS

CONIKS is a system that provides automated trust establishment with untrusted providers, allowing clients to verify their own consistency of key bindings. It uses merkle trees to represent information needed about individual users, and has security goals of having no 'spurious' keys and 'non-equivocation;. Multiple protocols, such as the Monitoring and Auditing Protocol, are implemented to check whether identity providers are lying or not. The secure communication model involves periodically checking consistency of key bindings and encrypting messages after verifying consistency.



CONIKS: Bringing key transparency to end users

End to End encryption Using public keys Problem with E2EE - key management is difficult for users If we query a provider to supply the public key, we need to trust the provider, which we cannot do

CONIKS provides automated trust establishment with untrusted providres clients can verify their own consistency of key bindings Security goal is makign provider key equivocation easily detectible

> Parties:
>
> - Identity providers, managing namespaces
> - clients
> - auditors

Registering a Key:

- Alice registers her key with her identity provider

Learning a User's key

- Bob is able to learn alice's key by first receiving it from the identity provider
- Then verifies that it is consistent using the STR of the merkle tree

Coniks uses merkle tree to allow clients to verify consistency of name-key bindings

Leaf node reperesents all information needed about an individual user $h_{leaf} = H(k_{leaf}, k_n, i, l, commit(name_i | keys_i))$

Security goals:

- No Spurious keys
- Non Equivocation

Multiple protocols

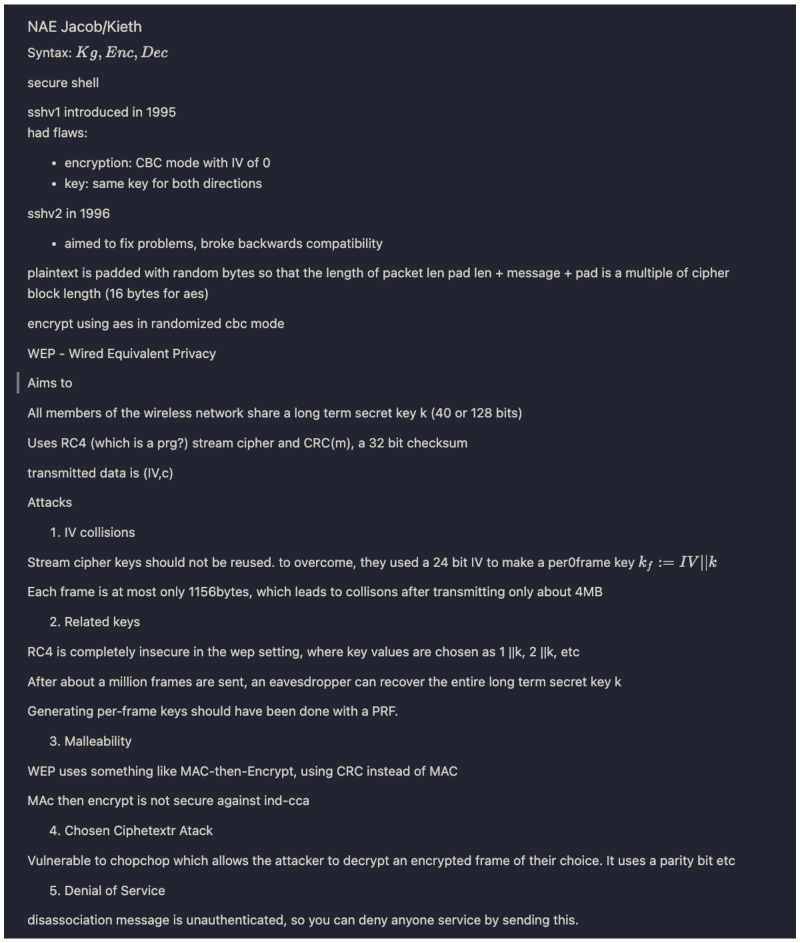
> - Monitoring Protocol
> - Auditing Protocol
>   - Auditors check whether id provider is lying or not
>   - Verifieable linear chain of STR history
>   - check if there is branching
>   - cross verification with mulitple providers

Secure conmmunication model with coniks

1. preiodically, client b checks consistentcy of bob's binding
2. before sending bobs message to client A, B looks up the public key for the username alice at foo.com
3. if client B determines that laice's binding is consistent, it encrypts and sends

NAE

The NAE presentation discussed the security protocols SSHv1 and SSHv2, as well as the Wired Equivalent Privacy (WEP) protocol used in wireless networks. SSHv1 had flaws with its encryption and key, while SSHv2 aimed to fix these issues but broke backwards compatibility. WEP is designed to make wireless networks as secure as wired networks by sharing a long-term secret key between all members, using RC4 stream cipher and CRC(m) 32-bit checksum. However, WEP is vulnerable to several attacks such as IV collisions, related keys, malleability, chosen ciphertext attacks, and denial of service attacks. Also, per-frame keys should have been generated using a pseudorandom function (PRF), rather than the current method.



NAE Jacob/Kieth

Syntax: Kg, Enc, Dec

secure shell

sshv1 introduced in 1995
had flaws:

- encryption: CBC mode with IV of 0
- key: same key for both directions

sshv2 in 1996

- aimed to fix problems, broke backwards compatibility

plaintext is padded with random bytes so that the length of packet len pad len + message + pad is a multiple of cipher block length (16 bytes for aes)

encrypt using aes in randomized cbc mode

WEP - Wired Equivalent Privacy

Aims to

All members of the wireless network share a long term secret key k (40 or 128 bits)

Uses RC4 (which is a prg?) stream cipher and CRC(m), a 32 bit checksum

transmitted data is (IV,c)

Attacks

1. IV collisions

Stream cipher keys should not be reused. to overcome, they used a 24 bit IV to make a per0frame key $k_f := IV||k$

Each frame is at most only 1156bytes, which leads to collisons after transmitting only about 4MB

2. Related keys

RC4 is completely insecure in the wep setting, where key values are chosen as 1 ||k, 2 ||k, etc

After about a million frames are sent, an eavesdropper can recover the entire long term secret key k

Generating per-frame keys should have been done with a PRF.

3. Malleability

WEP uses something like MAC-then-Encrypt, using CRC instead of MAC

MAc then encrypt is not secure against ind-cca

4. Chosen Ciphetextr Atack

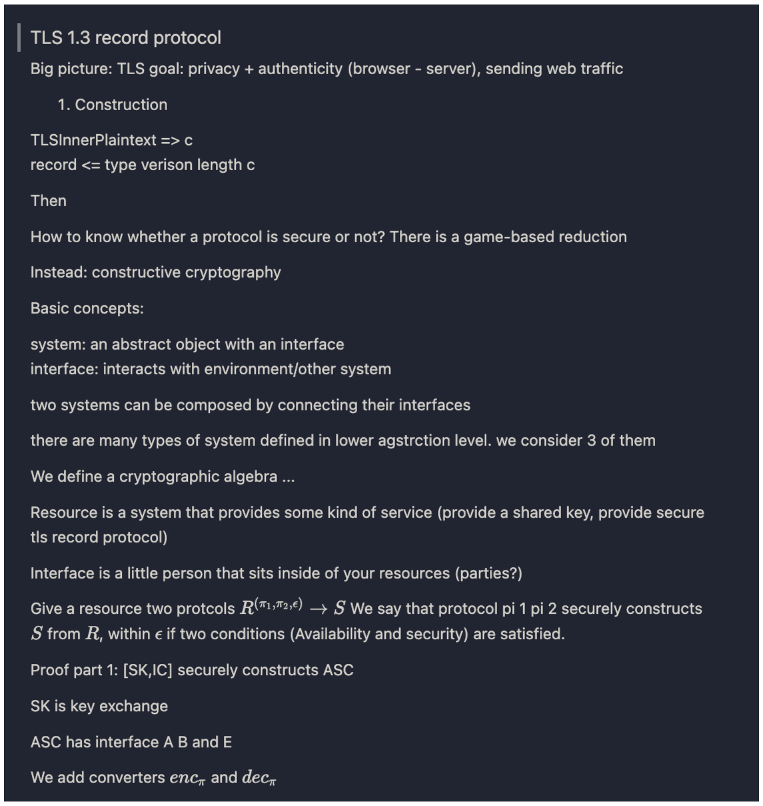
Vulnerable to chopchop which allows the attacker to decrypt an encrypted frame of their choice. It uses a parity bit etc

5. Denial of Service

disassociation message is unauthenticated, so you can deny anyone service by sending this.

TLS record protocol

This one was very confusing.



TLS 1.3 record protocol

Big picture: TLS goal: privacy + authenticity (browser - server), sending web traffic

1. Construction

TLSInnerPlaintext => c
record <= type verison length c

Then

How to know whether a protocol is secure or not? There is a game-based reduction

Instead: constructive cryptography

Basic concepts:

system: an abstract object with an interface
interface: interacts with environment/other system

two systems can be composed by connecting their interfaces

there are many types of system defined in lower agstrction level. we consider 3 of them

We define a cryptographic algebra ...

Resource is a system that provides some kind of service (provide a shared key, provide secure tls record protocol)

Interface is a little person that sits inside of your resources (parties?)

Give a resource two protcols $R^{(\pi_1, \pi_2, \epsilon)} \to S$ We say that protocol pi 1 pi 2 securely constructs S from R, within ϵ if two conditions (Availability and security) are satisfied.

Proof part 1: [SK,IC] securely constructs ASC

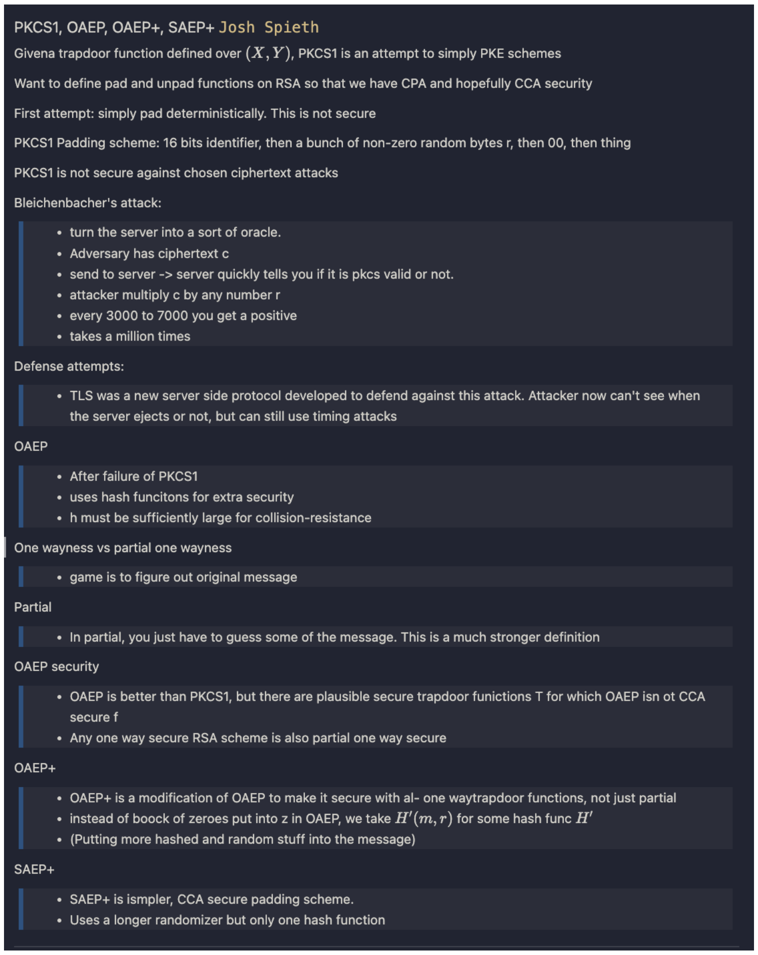
SK is key exchange

ASC has interface A B and E

We add converters enc_π and dec_π

PKCS1, OAEP, OAEP+, SAEP+

PKCS1 is a padding scheme designed for public-key encryption (PKE) that is not secure against chosen ciphertext attacks. Bleichenbacher's attack can be used to break PKCS1. OAEP was developed as a response to the failure of PKCS1 and uses hash functions to provide extra security. A trapdoor function T is partially one way if guessing some of the message is much easier than guessing the entire message. OAEP is better than PKCS1, but there exist trapdoor functions T for which OAEP is not chosen ciphertext attack (CCA) secure. Any one-way secure RSA scheme is also partially one-way secure. OAEP+ is a modification of OAEP that makes it secure with all one-way trapdoor functions, not just partially one-way trapdoor functions. SAEP+ is a simpler CCA-secure padding scheme that uses a longer randomizer but only one hash function.



PKCS1, OAEP, OAEP+, SAEP+ Josh Spieth

Givena trapdoor function defined over (X, Y), PKCS1 is an attempt to simply PKE schemes

Want to define pad and unpad functions on RSA so that we have CPA and hopefully CCA security

First attempt: simply pad deterministically. This is not secure

PKCS1 Padding scheme: 16 bits identifier, then a bunch of non-zero random bytes r, then 00, then thing

PKCS1 is not secure against chosen ciphertext attacks

Bleichenbacher's attack:

> - turn the server into a sort of oracle.
> - Adversary has ciphertext c
> - send to server -> server quickly tells you if it is pkcs valid or not.
> - attacker multiply c by any number r
> - every 3000 to 7000 you get a positive
> - takes a million times

Defense attempts:

> - TLS was a new server side protocol developed to defend against this attack. Attacker now can't see when the server ejects or not, but can still use timing attacks

OAEP

> - After failure of PKCS1
> - uses hash funcitons for extra security
> - h must be sufficiently large for collision-resistance

One wayness vs partial one wayness

> - game is to figure out original message

Partial

> - In partial, you just have to guess some of the message. This is a much stronger definition

OAEP security

> - OAEP is better than PKCS1, but there are plausible secure trapdoor funictions T for which OAEP isn ot CCA secure f
> - Any one way secure RSA scheme is also partial one way secure

OAEP+

> - OAEP+ is a modification of OAEP to make it secure with al- one waytrapdoor functions, not just partial
> - instead of boock of zeroes put into z in OAEP, we take $H'(m, r)$ for some hash func H'
> - (Putting more hashed and random stuff into the message)

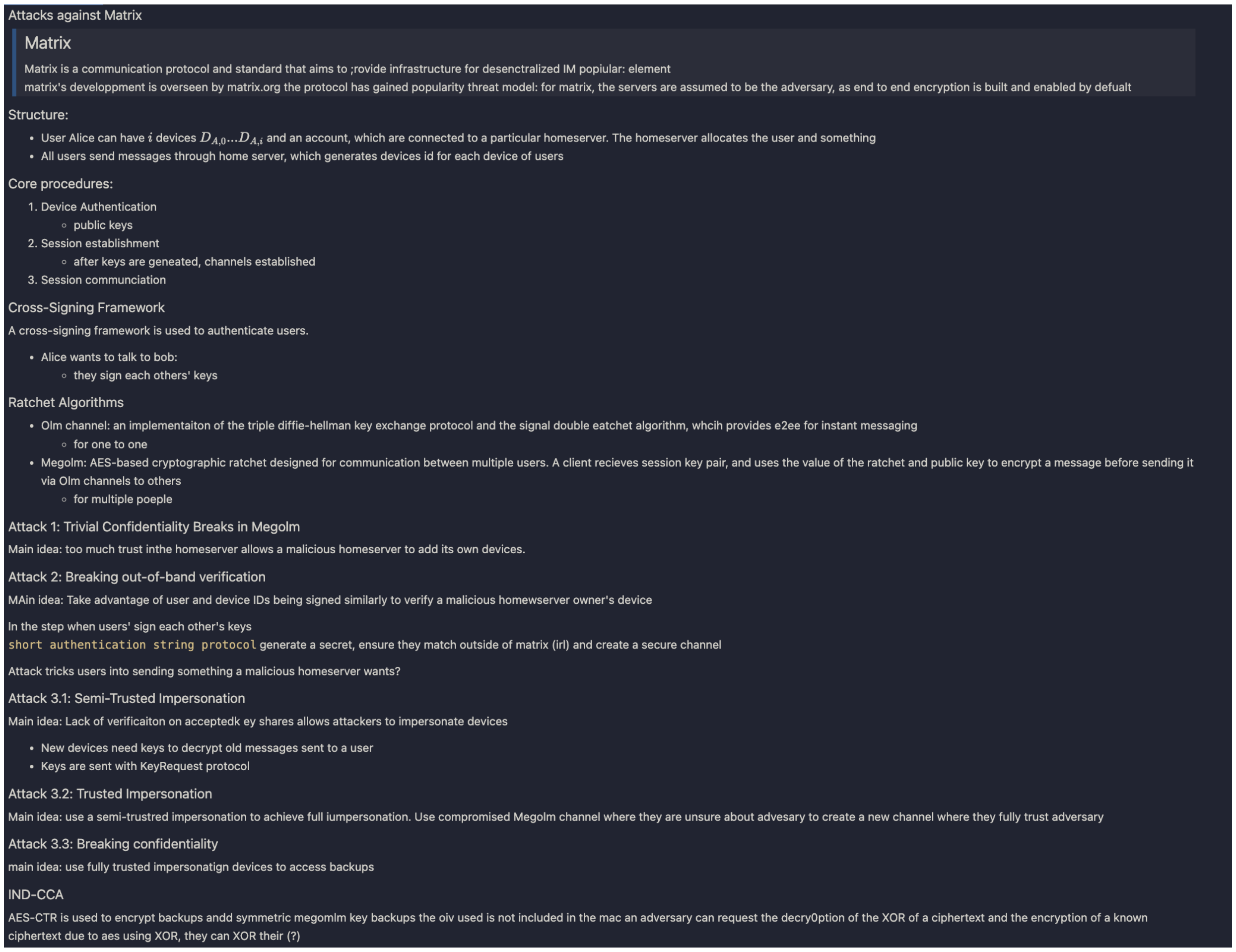
SAEP+

> - SAEP+ is ismpler, CCA secure padding scheme.
> - Uses a longer randomizer but only one hash function

Attacks against Matrix

This presentation discussed attacks against the Matrix communication protocol and standard, which provides infrastructure for decentralized instant messaging through the use of servers, homeservers, and end-to-end encryption. Users can have multiple devices and are authenticated through some framework. The protocol utilizes the Olm and Megolm ratchet algorithms for one-to-one and group communication, respectively.

Attack 1 targets Megolm and exploits trust in homeservers, allowing a malicious server to add its own devices. Attack 2 exploits the similarity in signing user and device IDs to perform a man-in-the-middle attack during the authentication process. Attack 3 is broken down into three sub-attacks. Attack 3.1 involves a lack of verification on accepted key shares, allowing attackers to impersonate devices. Attack 3.2 uses a semi-trusted impersonation to achieve full impersonation, and Attack 3.3 uses fully trusted impersonating devices to access backups.



Attacks against Matrix

> **Matrix**
>
> Matrix is a communication protocol and standard that aims to ;rovide infrastructure for desenctralized IM popiular: element
> matrix's developpment is overseen by matrix.org the protocol has gained popularity threat model: for matrix, the servers are assumed to be the adversary, as end to end encryption is built and enabled by defualt

Structure:

- User Alice can have i devices $D_{A,0}...D_{A,i}$ and an account, which are connected to a particular homeserver. The homeserver allocates the user and something
- All users send messages through home server, which generates devices id for each device of users

Core procedures:

1. Device Authentication
    - public keys
2. Session establishment
    - after keys are geneated, channels established
3. Session communciation

Cross-Signing Framework

A cross-signing framework is used to authenticate users.

- Alice wants to talk to bob:
    - they sign each others' keys

Ratchet Algorithms

- Olm channel: an implementaiton of the triple diffie-hellman key exchange protocol and the signal double eatchet algorithm, whcih provides e2ee for instant messaging
    - for one to one
- Megolm: AES-based cryptographic ratchet designed for communication between multiple users. A client recieves session key pair, and uses the value of the ratchet and public key to encrypt a message before sending it via Olm channels to others
    - for multiple poeple

Attack 1: Trivial Confidentiality Breaks in Megolm

Main idea: too much trust inthe homeserver allows a malicious homeserver to add its own devices.

Attack 2: Breaking out-of-band verification

MAin idea: Take advantage of user and device IDs being signed similarly to verify a malicious homewserver owner's device

In the step when users' sign each other's keys
`short authentication string protocol` generate a secret, ensure they match outside of matrix (irl) and create a secure channel

Attack tricks users into sending something a malicious homeserver wants?

Attack 3.1: Semi-Trusted Impersonation

Main idea: Lack of verificaiton on acceptedk ey shares allows attackers to impersonate devices

- New devices need keys to decrypt old messages sent to a user
- Keys are sent with KeyRequest protocol

Attack 3.2: Trusted Impersonation

Main idea: use a semi-trustred impersonation to achieve full iumpersonation. Use compromised Megolm channel where they are unsure about advesary to create a new channel where they fully trust adversary

Attack 3.3: Breaking confidentiality

main idea: use fully trusted impersonatign devices to access backups

IND-CCA

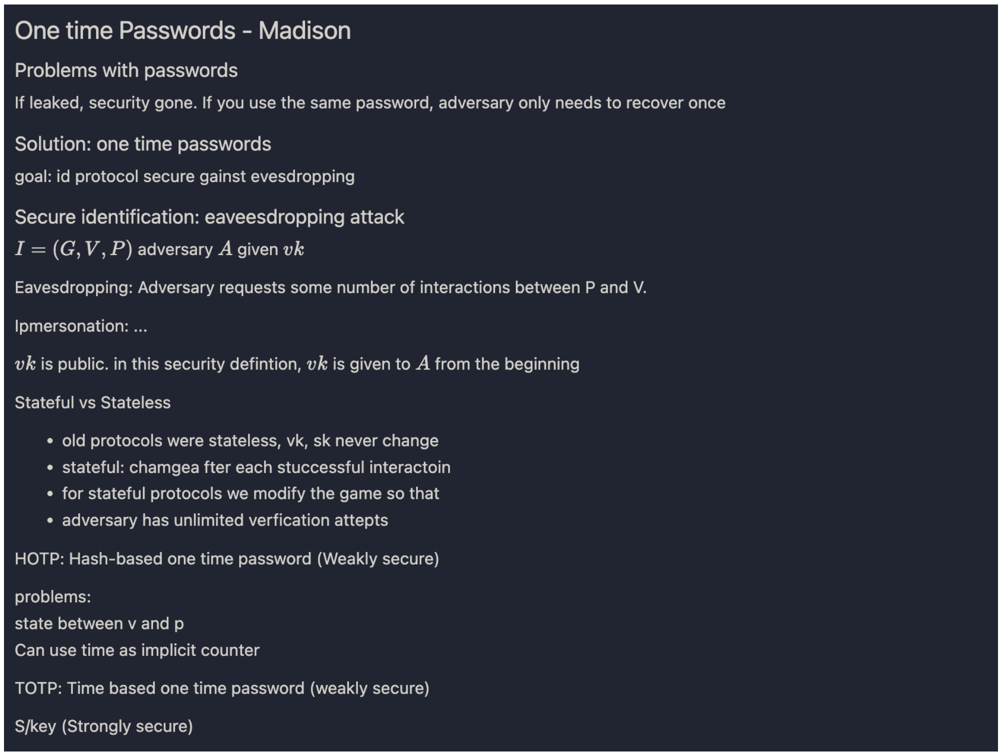
AES-CTR is used to encrypt backups andd symmetric megomlm key backups the oiv used is not included in the mac an adversary can request the decry0ption of the XOR of a ciphertext and the encryption of a known ciphertext due to aes using XOR, they can XOR their (?)

One time Passwords

One-time passwords (OTP) are a solution to the problem of using the same password for multiple authentications. The goal of OTP is to have a protocol that is secure against eavesdropping attacks. In the secure identification protocol, the adversary is given a verification key vk, and the eavesdropping attack involves the adversary requesting some interactions between the prover P and verifier V. The impersonation attack involves the adversary trying to impersonate P to V.

There are two types of OTPs: stateful and stateless. Stateless OTPs have a fixed verification key and secret key, while stateful OTPs change the keys after each successful interaction. HOTP (Hash-based One-Time Password) and TOTP (Time-based One-Time Password) are examples of weakly secure OTPs. S/Key is an example of a strongly secure OTP.

HOTP and TOTP use a counter or time to generate OTPs, while S/Key generates a sequence of OTPs from a secret key.



One time Passwords - Madison

Problems with passwords

If leaked, security gone. If you use the same password, adversary only needs to recover once

Solution: one time passwords

goal: id protocol secure gainst evesdropping

Secure identification: eaveesdropping attack

$I = (G, V, P)$ adversary A given vk

Eavesdropping: Adversary requests some number of interactions between P and V.

Ipmersonation: ...

vk is public. in this security defintion, vk is given to A from the beginning

Stateful vs Stateless

- old protocols were stateless, vk, sk never change
- stateful: chamgea fter each stuccessful interactoin
- for stateful protocols we modify the game so that
- adversary has unlimited verfication attepts

HOTP: Hash-based one time password (Weakly secure)

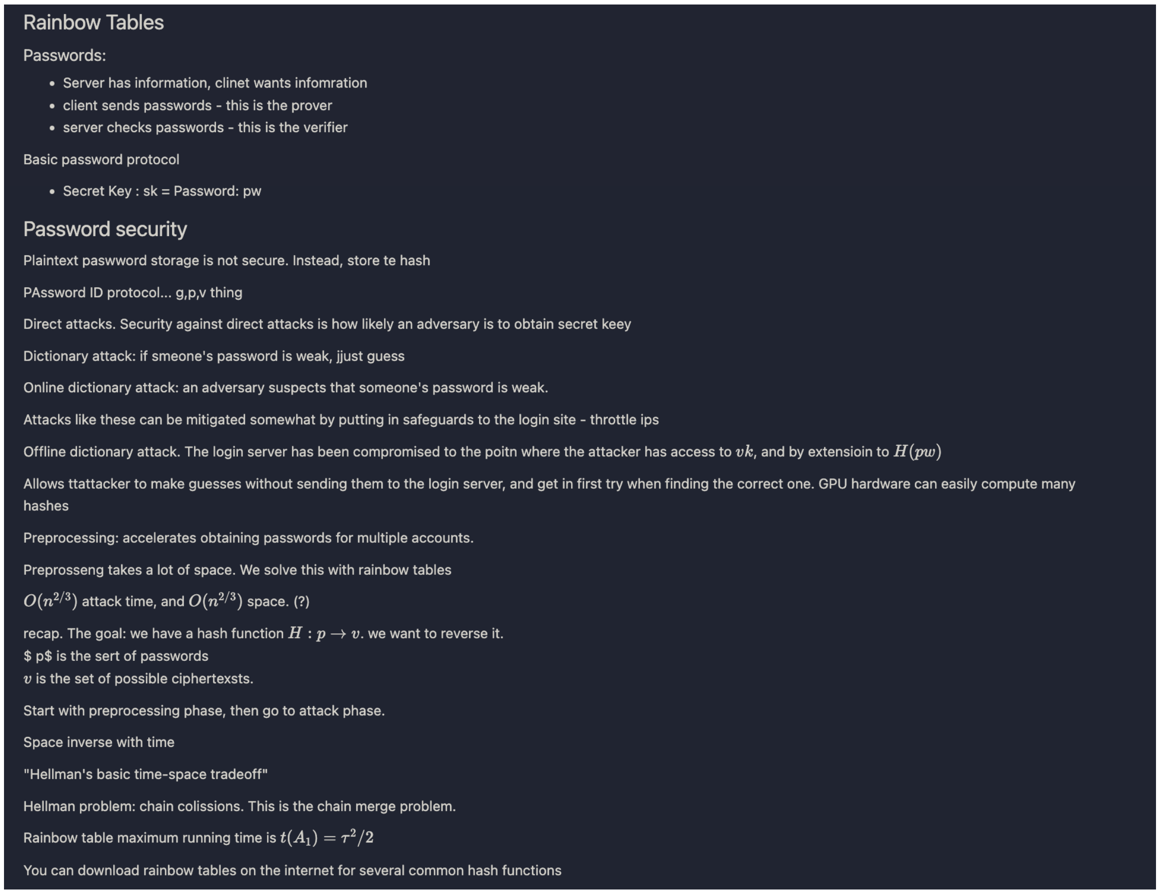
problems:
state between v and p
Can use time as implicit counter

TOTP: Time based one time password (weakly secure)

S/key (Strongly secure)

Rainbow Tables

Rainbow tables are a technique used to reverse hash functions by precomputing a table of chains. The goal is to find a password p given a hash value $v = H(p)$. The process involves two phases: preprocessing and attack. During the preprocessing phase, chains are computed and stored in a table. In the attack phase, the table is used to lookup a chain that ends in the target hash value and traverse it to find the corresponding password. Rainbow tables are designed to optimize storage space by using a reduced chain length and reducing the number of chains, while maintaining a tradeoff between the time and space complexity of the attack. The running time of a rainbow table attack is $O(n^{2/3})$, where n is the size of the password space. To prevent attacks, one can use saltst to make each user's hash unique and store the salted hash in place of the passwords. Strong passwords and rate limiting/throttling can help to protect against dict and online attacks.



Rainbow Tables

Passwords:

- Server has information, clinet wants infomration
- client sends passwords - this is the prover
- server checks passwords - this is the verifier

Basic password protocol

- Secret Key : sk = Password: pw

Password security

Plaintext paswword storage is not secure. Instead, store te hash

PAssword ID protocol... g,p,v thing

Direct attacks. Security against direct attacks is how likely an adversary is to obtain secret keey

Dictionary attack: if smeone's password is weak, jjust guess

Online dictionary attack: an adversary suspects that someone's password is weak.

Attacks like these can be mitigated somewhat by putting in safeguards to the login site - throttle ips

Offline dictionary attack. The login server has been compromised to the poitn where the attacker has access to vk, and by extensioin to $H(pw)$

Allows ttattacker to make guesses without sending them to the login server, and get in first try when finding the correct one. GPU hardware can easily compute many hashes

Preprocessing: accelerates obtaining passwords for multiple accounts.

Preprosseng takes a lot of space. We solve this with rainbow tables

$O(n^{2/3})$ attack time, and $O(n^{2/3})$ space. (?)

recap. The goal: we have a hash function $H : p \to v$. we want to reverse it.
$ p$ is the sert of passwords
v is the set of possible ciphertexsts.

Start with preprocessing phase, then go to attack phase.

Space inverse with time

"Hellman's basic time-space tradeoff"

Hellman problem: chain colissions. This is the chain merge problem.

Rainbow table maximum running time is $t(A_1) = \tau^2/2$

You can download rainbow tables on the internet for several common hash functions