Applied Crypto: Signal & Tor

USE SIGNAL!

?Use TOR?
Project 2...

- Project 2 is now live, some observations
- This is a **hard design** project
  - The actual coding is straightforward but getting the design right is really, really, really hard
- You are **not expected to get 100%**
  - This is part of the project's lesson: doing cryptographic systems right is really, really hard and you don't just lose 10%, but 100% on a mistake
- It is in go(lang) for several reasons:
  - Go gives really nice real world performance while being memory/typesafe
    - Especially for parallel programming
  - The learning curve is remarkably reasonable
  - It avoids the traps that other languages have on package/dependency management
Signal and Tor

• Signal is a messenger protocol and implementation
  • Signal (the company) is a 501(c)3 nonprofit
  • The protocol is also used by WhatsApp, Facebook Messenger, etc...

• Tor is an anonymity tool
  • Designed to provide anonymous but real-time network connectivity in the face of an aggressive but local adversary

• Common (bad) information security advice is "Use Signal, Use Tor"
  • In reality, Signal is a great protocol, but some security compromises are annoying in the implementation, so for most, WhatsApp is about as good
  • While Tor is often not just a placebo but poison!
End-To-End Messengers

- We love **end to end** cryptographic protocols...
- We love **forward secrecy**...
  - If someone steals our private keys, they can't recover old messages
  - After all, we want things to stay secret even if our keys are compromised
- Forward secrecy is "easy" for online protocols
  - Just make sure to do a DHE/ECDHE key exchange, and throw away the session key when done
- Forward secrecy is **much more annoying** for an offline protocol
  - Alice wants to share data with Bob, but Bob is **not online**
    - Like in project 2...
    - Or any messenger system!
Signal Requirements For Key Agreement

- Three parties: Alice, Bob, and a messenger server
  - The messenger server is like the file store in project 2, an untrusted entity
  - A separate mechanism is used to provide key transparency
- Bob is offline:
  - He has prearranged data stored on the messenger server
- Alice and Bob want to create an ephemeral (DH) key...
  - To use for then encrypting messages
- They need mutual authentication
  - Assuming Alice and Bob have the correct public keys, only Alice and Bob could have agreed on a key
- They also need deniability
  - Alice or Bob can't create a record proving the other side participated in creating the key: So no "Alice just signs her DH..." design
Extended Triple Diffie-Hellman

Key idea:
- Lets use multiple Diffie-Hellman exchanges combined into one
  - Some to perform mutual authentication
  - Some to generate an ephemeral key
  - Shove them ALL into a hash-based key derivation function

They use elliptic curves, but the design would be the same for conventional DH, so we will use the former
- We will use $DH(A,B)$ as $DH(g^a,g^b)$ where we know $a$ but not $b$.
  (So $A$ is our private value, $B$ is someone else's public value)
- Also have $Sign(K,M)$ for signing and $KDF(KM)$ which derives a bunch of session keys for a hash-based key derivation function (e.g. $PBKDF2$ with only a couple iterations)
Lots of Keys!

- All keys have both a public & private component
  - Private components always stay with Alice and Bob
  - Anything broadcast is always the public component

- Alice:
  - $IK_A$: Alice's identity key: for both DH and signatures
  - $EK_A$: Alice's ephemeral key: Created randomly just to talk to Bob.

- Bob:
  - $IK_B$: Bob's identity key, long lived
  - $SPK_B$: Bob's signed rekey, rotates ~weekly/monthly
    - Has corresponding signature $\text{Sign}(IK_B, SPK_B)$
  - $OPK_B$: Bob's one time use keys (One Time Prekey)
    - Can run out, but designed to increase security when available
Before We Start:
Bob to Server, Server to Alice

- Bob uploads:
  - $IK_B$, $SPK_B$, $Sign(IK_B, SPK_B)$, $\{OPK_{B1}, OPK_{B2}, OPK_{B3} \ldots\}$

- Now when Alice wants to talk to Bob...
  - Gets from the server:
    - $IK_B$, $SPK_B$, $Sign(IK_B, SPK_B)$, $OPK_B$?
    - Told which OPK it is or "There are no OPKs left"
      - OPKs are designed to prevent replay attacks:
        Bob will never allow any particular OPK to be used twice

- This is now the input into Alice's DH calculations
Alice now does a lot of DH...

- **$DH1 = DK(IK_A, SPK_B)$**
  - Acts as authentication for Alice when Bob does the same
- **$DH2 = DK(EK_A, IK_B)$**
  - Forces Bob to do mutual authentication
- **$DH3 = DK(EK_A, SPK_B)$**
  - Adds in ephemeral $EK_A$ to short lived $SPK_B$
- **$DH4 = DK(EK_A, OPK_B)$**
  - Adds in one-time used $OPK_B$, if available
- **$SK = HKDF(DH1 \parallel DH2 \parallel DH3 \parallel DH4)$**
  - Skip $DH4$ if no one time pre-keys are available
  - Now discard the private part of $EK_A$ and the intermediate DH calculations
HKDF...

- Hash Based Key Derivation Function...
  - AKA how to use HMAC to create several keys starting from a single key
- Why? Different keys for different purposes
  - Encryption keys in different directions, separate MAC keys
- Very simple construction

```python
hkdf(keydata, info, L):
    T = Out = ""
    for (i = 1; i <= ceiling(L/hashlen); ++i){
        T = HMAC(keydata, T || info || i);
        Out = Out || T
    }
    return Out[0:L-1]
```
Now Alice Sends To Bob

- \( IK_A, EK_A, \) which \( OPK \) used (if any), and \( E(SK, M, IK_A \| IK_B) \)

  - Using an AEAD encryption mode: 
    Authenticated Encryption with Additional Data modes allow additional data to be protected by the MAC but sent in the clear:
    In this case \( IK_A \) and \( IK_B \)

- Bob can do the same DH calculations to generate SK
  - Since Bob knows the private keys corresponding to the public values Alice used
  - If it fails to verify the AEAD data abort:
    How we know that \( IK_a \) and \( IK_b \) are sent honestly
Key Transparency

- For now, Alice and Bob are trusting the server to report $\text{IK}_A$ and $\text{IK}_B$ correctly
  - If the server lies, 🤷

- Fortunately there is an answer:
  If Alice and Bob are *ever* together:
  - One person's phone displays $H(\text{IK}_A \parallel \text{IK}_B)$ as a QR Code
  - Other person's phone verifies that it is the same

- Plus the voice channel...
  - Display "Two Words" on screen:
    $F(H(\text{IK}_A \parallel \text{IK}_B \parallel \text{SK}))$
  - Assumption is a MitM attacker can't fake a voice conversation quickly enough, so if each person says one of the words...
Considerations

- Authentication requires the out-of-channel methods
  - Otherwise no guarantees: Absent the out of channel the keyserver could be lying

- Replay attacks
  - Only if no OPK is available: Can be potentially bad

- Deniability
  - No cryptographic proofs available as to the sender/receiver!
  - So if Bob releases a message saying "Alice sent me X", Alice can go "Nope, never did" and Bob can't release anything proving that the message was created by Alice and not Bob:
    Both possess the cryptographic material necessary to create the message
And Then Ratchets...

- A "ratchet" is a one-way function for message keys
  - \( \text{Ratchet}(K_i) \rightarrow K_{i+1}, MK_i \)
  - But can't take \( K_{i+1} \) and \( MK_i \) to find \( K_i \)

- A symmetric key ratchet is easy
  - We've seen these already:
    Any secure PRNG with rollback resistance is a ratchet
  - Can do it slightly more efficiently with HMAC:
    \( \text{HMAC}(K_i, 0x01) \rightarrow MK_i \)
    \( \text{HMAC}(K_i, 0x02) \rightarrow K_{i+1} \)

- It's OK to keep around the intermediate session keys
  - Thanks to HMAC we can't go backwards with them anyway:
    Needed for out of order messages
Signal adds in DH ratchets too...

• So for a few messages in a chain you use a symmetric key ratchet...
  • You gain forward secrecy by discarding the old internal state
• But occasionally you rekey with an additional DH
  • Used to add into the ratchet internal state: update $K_i$ to $H(K_{i-1} \| DH)$
• Acts to reset everything with even more randomness
  • So even if you compromise Bob's device at time $T$ and steal all the keys...
  • You can't decrypt old messages that aren't on Bob's device: can't run the symmetric ratchet backwards
  • You can't decrypt subsequent messages once Bob & Alice use a DH ratchet
The Protocol is Great...

BUT!

• The app itself does some ehh thing in the usability/security tradeoff...
  
  • *No mechanism to back-up messages!*  
  If your phone is toast, your messages are gone!
  
  • *No mechanism to migrate to a new phone!*  
  If you upgrade to a new phone, your messages are gone!
  
  • *Auto-notifies all those where you are in their contacts that they join*

• This is where WhatsApp has a huge competitive advantage
  
  • They allow backup of messages, message migration etc...
And A Particular Problem: Naming/Identifying People...

- How does Alice identify Bob in a system? How does Bob register his keys for the first time?
  - Name? There are lots of people named Bob!
  - Email? Email addresses don't tend to be the most secure thing in the world...

- Signal's solution: phone #
  - Phone numbers are a lot harder to hijack than email addresses

- But this creates a problem: Not everyone wants to reveal their phone #
And Signal Makes It Worse...

- When you register your phone # with Signal...
- It broadcasts to everyone who has you in their contacts that you are now on Signal
  - And with no notice or control to you...
- You think this might be a problem? Because I think this is a problem...
  - Phone # is a lot more disruptive information in the hands of an abuser than an email address is...
Tor: The Onion Router
Anonymous Websurfing

- Tor actually encompasses many different components

- The Tor network:
  - Provides a means for anonymous Internet connections with low(ish) latency by relaying connections through multiple Onion Router systems

- The Tor Browser bundle:
  - A copy of FireFox extended release with privacy optimizations, configured to only use the Tor network

- Tor Hidden Services:
  - Services only reachable though the Tor network

- Tor bridges with pluggable transports:
  - Systems to reach the Tor network using encapsulation to evade censorship

- Tor provides three separate capabilities in one package:
  - Client anonymity, censorship resistance, server anonymity
The Tor Threat Model: Anonymity of content against local adversaries

- The goal is to enable users to connect to other systems “anonymously” but with low latency
  - The remote system should have no way of knowing the IP address originating traffic
  - The local network should have no way of knowing the remote IP address the local user is contacting

- Important what is excluded: The global adversary
  - Tor does not even attempt to counter someone who can see all network traffic: It is probably impossible to do so and be low latency & efficient
The High Level Approach: Onion Routing

- The Tor network consists of thousands of independent Tor nodes, or “Onion Routers”
  - Each node has a distinct public key and communicates with other nodes over TLS connections
- A Tor circuit encrypts the data in a series of layers
  - Each hop away from the client removes a layer of encryption
  - Each hop towards the client adds a layer of encryption
- During circuit establishment, the client establishes a session key with the first hop…
  - And then with the second hop through the first hop
- The client has a *global* view of the Tor Network:
  The directory servers provide a list of all Tor relays and *their public keys*
Tor Routing
In Action
Tor Routing
In Action
Creating the Circuit Layers...

• The client starts out by using an authenticated DHE key exchange with the first node...
  • OR1 creates \( g^a \), signs it with its private key, sends \( g^a, \text{Sign}(K_{\text{priv}_{\text{or1}}, g^a}) \) to client
  Client creates \( g^b \), sends it to OR1
  Client does \( \text{Verify}(K_{\text{pub}_{\text{or1}}, g^a}) \)
  • Creating a session key \( K_{OR1} \) as \( H(g^{ab}) \)
    • This first hop is commonly referred to as the “guard node”

• It then tells OR1 to extend this circuit to OR2
  • Through that, creating a session key for the client to talk to OR2 that OR1 \textit{does not know}
  • And OR2 doesn’t know what the client is, just that it is somebody talking to OR1 requesting to extend the connection...

• It then tells OR2 to extend to OR3…
  • And OR1 won’t know where the client is extending the circuit to, only OR2 will
Unwrapping the Onion

• Now the client sends some data…
  • \( E(K_{or1}, E(K_{or2}, E(K_{or3}, Data))) \)
  • OR1 decrypts it and passes on to OR2
    • \( E(K_{or2}, E(K_{or3}, Data)) \)
  • OR2 then passes it on…
• Generally go through at least 3 hops…
  • Why 3? So that OR1 can’t call up OR2 and link everything trivially
• Messages are a fixed-sized payload
The Tor Browser…

• Surfing “anonymously” doesn’t simply depend on hiding your connection…
• But also configuring the browser to make sure it resists tracking
  • No persistent cookies or other data stores
  • No deviations from other people running the same browser
• Anonymity only works in a crowd…
  • So it really tries to make it all the same
• But by default it makes it easy to say “this person is using Tor”
But You Are Relying On Honest Exit Nodes…

- The exit node, where your traffic goes to the general Internet, is a man-in-the-middle…
  - Who can see and modify all non-encrypted traffic
  - The exit node also does the DNS lookups
- Exit nodes have not always been honest…
Anonymity Invites Abuse…
(Stolen from Penny Arcade)
This Makes Using Tor Browser Painful...
And Also Makes Running Exit Nodes Painful…

- If you want to receive abuse complaints…
  - Run a Tor Exit Node

- Assuming your ISP even allows it…
  - Since they don’t like complaints either

- Serves as a large limit on Tor in practice:
  - Internal bandwidth is plentiful, but exit node bandwidth is restricted

- Know a colleague who ran an exit node for research…
  - And got a *visit from the FBI!*
Censorship Evasion...

- Tor is actually really **bad** for evading censorship
  - It is trivial to tell that someone on the network is running Tor
- There are **optional** pluggable transports that attempt to hide the traffic
  - The problem is you have to learn about these...
    - Yet if the censor does, it won't work!
- And then the user has all the bad of Tor...
  - Fate sharing with the exit nodes
  - Significantly worse latency
  - Oh, and Tor Browser's not saving history is not necessarily nice!
- Only good thing is it is "free"
  - Tor project gets paid largely for counter-censorship
  - Users are "paying" by providing traffic for those who want anonymity to hide in
Tor Browser is also used to access Tor Hidden Services aka .onion sites

- Services that **only** exist in the Tor network
  - So the service, not just the client, has possible anonymity protection
  - The “Dark Web”

- A **hash** of the hidden service's public key
  - http://pwoah7foa6au2pul.onion
    - AlphaBay, one of many dark markets, now deceased
  - https://facebookcorewwwi.onion
    - In this case, Facebook spent a lot of CPU time to create something distinctive
      (Also a proof of work that Facebook spent a huge amount of time generating private keys to find one where the public key’s hash started with "Facebook" and the rest sort of made sense)

- Using this key hash, can query to set up a circuit to create a hidden service at a rendezvous point
  - And because it is the hash of the key we have end-to-end security when we finally create a final connection
Tor Hidden Service:
Setting Up Introduction Point
Tor Hidden Service:
Query for Introduction, Arrange Rendezvous
Tor Hidden Service: Rendevous and Data
We highly recommend that you disable Javascript when viewing the marketplace for better security.
Remarks...

- A hidden service wants to keep the guard node constant for a long period of time...
  - Since the creation of new circuits is far easier to notice than any other activity
- Want to use a different node for the rendezvous point and introduction
  - Don’t want the rendezvous point to know who you are connecting to
- These are slow!
  - Going through 6+ hops in the Tor network!
Non-Hidden Tor Hidden Service: Connect Directly to Rendezvous
Non-Hidden Hidden Services
Improve Performance

- No longer rely on exit nodes being honest
  - No longer rely on exit node bandwidth either
- Reduces the number of hops to be the same as a not hidden service
- Result: Huge performance win!
  - Not slow like a hidden service
  - Not limited by exit node bandwidth
  - Facebook does this
- Any legitimate site offering a Tor hidden service should use this technique
  - Since legitimate sites don't need to hide!
Real use for *true hidden* hidden services

- "Non-arbitrageable criminal activity"
- Some crime which is universally attacked and targeted
  - So can't use "bulletproof hosting", CDNs like CloudFlare, or suitable “foreign” machine rooms:
    And since CloudFlare will service the anti-Semitic shitheads like gab.ai and took forever to get rid of the actual nazis of Stormfront and the murderous shits of 8chan...

- Dark Markets
  - Marketplaces based on Bitcoin or other alternate currency

- Cybercrime Forums
  - Hoping to protect users/administrators from the fate of earlier markets

- And worse...
The Dark Market Concept

- Four innovations:
  - A censorship-resistant payment (Bitcoin)
    - Needed because illegal goods are not supported by Paypal etc
      - Bitcoin/cryptocurrency is the only game in town for US/Western Europe after the Feds smacked down Liberty Reserve and eGold
  - An eBay-style ratings system with mandatory feedback
    - Vendors gain positive reputation through continued transactions
  - An escrow service to handle disputes
    - Result is the user (should) only need to trust the market, not the vendors
  - Accessable only as a Tor hidden service
    - Hiding the market from law enforcement
The Dark Markets: History

- All pretty much follow the template of the original “Silk Road”
- Founded in 2011, Ross Ulbricht busted in October 2013
- The original Silk Road actually (mostly) lived up to its libertarian ideals
  - Including the libertarian ideal that if someone rips you off you should be able to call up the Hell’s Angels and put a hit on them
  - And the libertarian idea if someone is foolish enough to THINK you are a member of the Hell’s Angels you can rip them off for a large fortune for a fake hit
- Since then, markets come and go...
  - And even information about them is harder:
    - Reddit no longer supports them, deepdotweb got busted...
    - Leaving "Dread": Reddit as a Tor Hidden Service
The Dark Markets: Not So Big, and *Not Growing!*

- Kyle Soska and Nicolas Christin of CMU have crawled the dark markets for years
  - These markets *deliberately* leak sales rate information from mandatory reviews
  - So simply crawl the markets, see the prices, see the volume, voila…

**Takeaways:**
- Market size has been relatively steady for years, about $300-500k a day sales
  - Latest peak got close to $1M a day
- Dominated by Pot, MDMA, and stimulants, with secondary significance with opioids and psychedelics
- A few sellers and a few markets dominate the revenue: A fair bit of “Winner take all”
  - But knock down any “winner” and another one takes its place
The Scams…

- You need a reputation for honesty to be a good crook
  - But you can burn that reputation for short-term profit
- The “Exit Scam” (e.g. pioneered by Tony76 on Silk Road)
  - Built up a positive reputation
  - Then have a big 4/20 sale
  - Require buyers to “Finalize Early”
    - Bypass escrow because of “problems”
    - Take the money and run!
- Can also do this on an entire market basis
  - The “Sheep Marketplace” being the most famous
And Now A Content Warning...

- The rest of the lecture is going to talk about the Elephant in the Room with Tor...
  Tor hidden services facilitate child abuse on an industrial scale
- And the Tor project **DOES NOT CARE**!
- I will be talking about actual cases and the scope of the problem
  - I studied these cases because they touched on significant policy issues surrounding searches and government hacking
- This will not be on the test beyond the following: "Yes, Nick does hate Tor with the fires of a thousand suns" and this is why...
  - And for the love of everything do not ever build something that has proved as loathsome as Tor
February 2, 2020, Sunrise, Florida

- A team of FBI agents in the Violent Crimes Against Children division, including special agents Daniel Alfin and Laura Schwartzenberger, attempted to serve a search warrant as part of a CSAM (Child Sexual Abuse Material) investigation.

- Agents Alfin and Schwartzenberger were murdered by the suspect and three other agents injured.

- I knew Dan professionally from his previous work involving CSAM and Tor...
The "Playpen" Investigation

- In 2015 the FBI managed to identify and capture the server hosting the "Playpen" child exploitation site: Daniel Alfin was one of the lead investigators.
- Playpen operated as a hidden service image board for posting CSAM:
  - 250,000+ registered users, 20,000+ images
  - This represents thousands of abused children!
- But the site operator's are not the only problem...
The site users are a problem:
  - A significant number are "hands-on" abusers:
    Both because of their predilections and because creating new "content" is currency in these communities.
To Deanonymize the Users...

- The FBI took over Playpen and ran the site for 2 weeks
- During those two weeks...
  - Disabled posting of new content, but continued to serve old content...
  - And added a post-login bonus: A zero-day attack on the Tor Browser Bundle
- Exploit payload: "phone home"
  - Not a general purpose shellcode, instead collect Ethernet Addresses, current user, and similar identifying information and contact an FBI server
- FBI calls this a NIT: "Network Investigation Technique"
- They had a warrant:
  - It described with particularity what it would search for, how it would work conceptually, etc...
Significant Impact

• 25 producers prosecuted, 350 arrests in the US alone
• Nearly 300 children identified or rescued from abusive situations worldwide, over 50 in the US
• But also two significant controversies:
  • Was the warrant actually valid?
    • Answer ended up being "No, but 'good faith'....":
      At the time there was no way to write a warrant that says "I want to search these computers, but we don't know where they are!"
  • What should defendants be able to examine with regard to the exploit?
    • Answer largely ended up being "No, not actually relevant"
    • An in the weeds discussion by Susan Hennesey and myself is available here:
The Problem: These are communities of abusers

- There have been others both before and since
  - Before Playpen there was "Freedom Hosting": hosted close to 50 CSAM sites. If you want to be nauseated read the Freedom Hosting NIT warrant application
    - But "Freedom Hosting" they simply replaced the content with a "doing maintenance" page where the NIT was quickly spotted
  - In 2017 an FBI style NIT was deployed on "GiftBox" (probably by the French): But it was captured by a site user and posted to Reddit...
  - In 2018 "Welcome to Video" was busted: Pay for CSAM with Bitcoin! Again, if you want to vomit read the indictments

- Communities create dangerous cycles of normalization
  - And larger communities are more dangerous: See more mild versions that happened on Reddit with TheDonald, jailbait, creepshots, etc...
    - Self reinforcement behavior: "Its normal because others in the community do it" and the community becomes self justifying
    - See the "Jailbait" analysis in Twitter and Tear Gas
  - Drives to extremes: Over the past decade, the age of CSAM victims has basically gotten younger... To the point where average age really can't get much lower
The Problem #2: The Tor Project **JUST DOES NOT CARE!**

- They treat this as "collateral damage" with a series of excuses. Here are actual justifications by Roger Dingledine (Founder):
  - "But hidden services are in their infancy"
    - And in the same presentation talk about it being a 10 year old idea...
  - "But hidden services are end-to-end authenticated"
    - Yeah, there is this thing call TLS...
  - "But hidden services work through NATs"
    - Yeah, there is this thing called uPNP: You ask the NAT to allow inbound connections
    - Oh, or just use EC2...
  - "But dissidents..."
    - Well, running Tor is very noticeable...
    - Plus you can "arbitrage host": Want to piss off China? Host in the US. Piss off the US? Host in Russia...
  - "But Facebook/SecureDrop/Etc... has an onion service"
    - Uh, they don't actually need to be hidden! And work better when they aren't!
And A Different Problem: Grooming

- I never encountered Agent Schwartzenberger, but this was her specialty...
  people who use electronic chat to groom child victims for exploitation

- In unencrypted chats, the chat-provider can *theoretically* try to detect this behavior
  - A case where classic Machine Learning tends to work pretty well if the results are human-reviewed for false-positives

- The problem grows even harder when dealing with encrypted chats
  - Since there is no longer a central server that can try to detect the behavior...
  - And the developers would probably resist adding an AI-snitch to the client