Spring 5-12-2023

# Going Dark and Encryption

Brendan Funk
*La Salle University*, funkb1@lasalle.edu

# Going Dark and Encryption

Brendan Funk

La Salle University

## *Introduction*

Law officers across the country and around the world are being left in the technological dust by their criminal counterparts. They have no problem obtaining evidence, however they run into issues accessing this information due to various encryption techniques being used. This phenomenon has been dubbed the "Going Dark" problem. James Comey describes the Going Dark problem as, "We have the legal authority to intercept and access communications and information pursuant to court order, but we often lack the technical ability to do so" (Comey, 2014).

The Going Dark problem is a relatively new problem facing law enforcement officers (LEOs) that has roots going back to the Crypto Wars of the early 1990s. At its core, the Going Dark problem is really just an issue of how to attack encrypted data. Data is either at rest, or in motion, and can be attacked several different ways depending on which state it is in. Recently, the FBI has found some success using a man-in-the-middle attack on criminals' cell phones, but since they sold the cell phones themselves, they were able to attack data both at rest and in motion. Today, LEOs are trying to solve the Going Dark problem by attacking encrypted data using a variety of tactics, and by trying to amend the Communications Assistance to Law Enforcement Act (CALEA) to include email and social media.

First, this paper will discuss the Crypto Wars, then data at rest and ways to attack it. Next, amending CALEA, data in motion and ways to attack data in motion will be discussed. Finally, this paper will discuss Anom and other possible solutions to the Going Dark problem.

## *Crypto Wars*

As the saying goes, everything old is new again. The Going Dark problem is just the modern-day version of a problem that has faced people for generations, and that problem is: "How do I keep my information and messages secure while still being able to read theirs?" Of course, today the question is more, "How do I encrypt my data that is at rest or in motion while being able to decrypt theirs?" The roots of this modern-day problem can be traced back to the 1970s with the onset of the so-called "Crypto Wars" (Manpearl, 2017). Before that discussion begins, it is important to define some terms. Encryption is defined as "the process of encoding data or information such that only those who are authorized by the creator of the information are able to access the data or information" (Manpearl, 2017). Data at rest is exactly what it sounds like, it is simply data that is not moving from one part of cyberspace to another. And finally, data in motion is data that is moving from one place to another.

The Crypto Wars began in the 1970s, but they did not really kick into gear until the 1990s. By then, computers had finally gotten to the point where they were cheap enough for the general public to afford and had enough processing power to securely transmit information (Manpearl, 2017). The government began to really worry about their ability to monitor and decrypt this information. Before the 1990s, computers were far too big and far too expensive for the average person to afford. Now, however, the playing field was beginning to level.

Two of the ways the government tried to combat the new computers were export control and something called key escrow, which was basically where either the government or a third party would hold the decryption keys in escrow (Manpearl, 2017). Export control had been the initial strategy in the Crypto Wars and it worked by doing two things: restricting technology companies from exporting encryption technology outside of the US, and the government placed prohibitions on researchers to prevent them from publishing any research in cryptography

(Christie, 2019). The controversial key escrow system the government came up with was called the Clipper Chip and was developed by the NSA in 1993 (Manpearl, 2017).

The Clipper Chip was a bad idea that failed even more miserably in practice. It was designed to provide encryption for phones and produced an encryption key that was available to the NSA. The federal government made it voluntary to implement, however it was the federal standard and so they thought it would be picked up by the general public pretty quickly (Manpearl, 2017). Almost as soon as the Clipper Chip made its way into the market, it was cracked. A cryptographer named Matt Blaze was able to build a program with a valid looking but invalid checksum that fooled the Clipper Chip (Blaze, 1994). This quickly led to the Clipper Chip program ending. While Clipper Chip was a failure, the government also attempted to prevent cryptographic research from leaving its shores.

The government has a list it maintains called the US Munitions List, and essentially if an item is on this list, then the exporting of such an item is heavily regulated by the Department of Defense (Manpearl, 2017). Most things on the list include what one would expect: guns and various forms of explosives. However, up until the early 1990s encryption software and related technology was also on this list (Manpearl, 2017). By the end of the 1990s, the government suffered another blow when the Ninth Circuit Court of Appeals ruled that their export controls violated the First Amendment rights of cryptographic researchers (Bernstein v. DOJ, 1999).

Another thing that worried the federal government (especially the FBI), was the shift of copper wiring to fiber optics. It was thought back then that fiber optics would lead to wiretaps becoming completely useless which was utterly incomprehensible to the FBI (Manpearl, 2017). The solution to this was to pass something called Communications Assistance for Law Enforcement Act (CALEA), which essentially required telecom companies to ensure that law

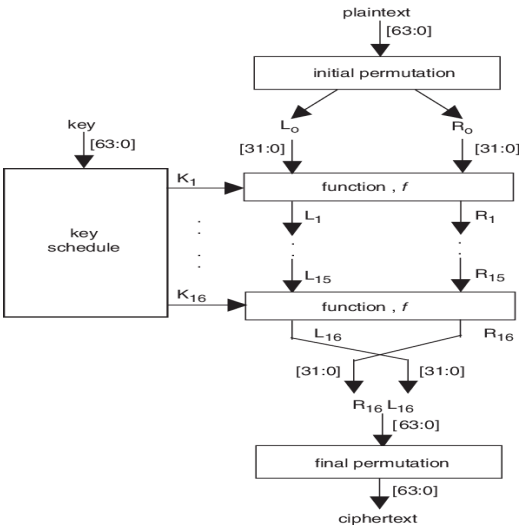
enforcement would be able to access the network if they had a warrant. This allowed the government to ensure that its employees would always be able to wiretap telephones for the foreseeable future. CALEA was eventually extended to all internet and Voice over Internet Protocol communications; however, CALEA did not address the elephant in the room that was (and remains) encryption. Further government attempts to deal with private encryption like the Patriot Act and FISA, ultimately failed to really solve the issue at hand. As the 1990s came to an end, so too did the Crypto Wars, with a pretty decisive victory in favor of private citizens. This would not prevent the government from continuing its attempts to get at information criminals were hiding.

### *Data at Rest*

One of the biggest failings that CALEA has is that it does not really deal with data at rest. Obviously, data that is able to be wiretapped is in motion, because it would be pretty useless to listen to a database or the cloud when they are not actively transmitting data. There are many encryption algorithms for data at rest, however three of the most popular are the Advanced Encryption Standard (AES), the Data Encryption Standard (DES), and Rivest-Shamir-Adleman (RSA). Today, the only ones out of those three really in use are AES and RSA, however it is important to understand all three in order to figure out ways for the government to solve this Going Dark problem.

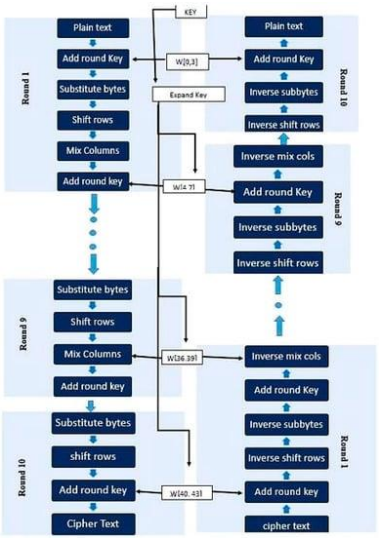
DES was actually adopted by the US government as their encryption algorithm in the 1970s and was the precursor to the modern-day AES (Grabbe, n.d.). DES works by splitting messages in two and performing different permutations on each half of the message. At the end, the messages are flipped so the second half is first and put through a final permutation to obtain the ciphertext (McLoone & McCanny, 2003). By 1998, John Gilmore and the Electronic Frontier

Foundation (EFF) built a computer that could, “go through the entire 56-bit DES key space in an average of 4.5 days” (Grabbe, n.d.). They actually cracked one code in 56 hours. Of course, the FBI responded with something called Triple DES. Triple DES is just DES done three times, but instead of having a key space of 56 bits, it has a key space of 2^112 (Grabbe, n.d.).



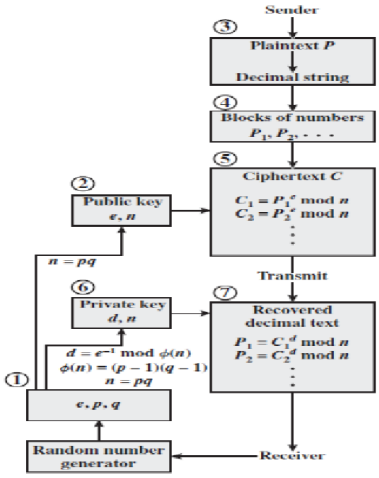

(McLoone & McCanny, 2003) Figure 1

The next evolution of DES was AES, and it is used by the US government as a replacement for DES. AES is part of a family of ciphers called Rijndael, and the biggest reason for switching from DES to AES was speed. AES is simply faster than DES without having to sacrifice security (Daoud and Huen, 2022). AES works by going through some n number of rounds to be encrypted. First, they add a round key, then the bytes are substituted, the rows are shifted, the columns are mixed, and they add the round key back at the end as well. Also, even for a 16-byte key, the number of applications of Rijndael to brute force it is 2^127, which is 15 orders of magnitude larger than triple DES. AES and DES are pretty similar in how they work, however in DES, only one half of the message is transformed in each round of encryption, where in AES the entire block is transformed in every round of encryption (Rijndael, 1999).




(Fatima, et al., 2022) Figure 2

RSA is different from DES and AES because it is an asymmetric algorithm instead of a symmetric algorithm. The difference is that asymmetric algorithms have individualized private keys for decoding, while symmetric algorithms have a single private key (Fatima, 2022). The basic process of RSA relies on the factoring problem, which says that there is no algorithm that is able to efficiently factor an arbitrarily given integer. RSA works by generating three random numbers e, p and q. A public key is generated with e and n. N is calculated by multiplying p * q. While this is going on the sender is sending a plaintext message with a decimal string that is broken down into blocks of numbers and then ciphertext. This ciphertext is created by raising each block P and raising it to the power of e, then the modulus n of P^e. Finally, it is decrypted by raising each block of ciphertext C to the power of d then taking the modulus n. RSA is a lot more popular in the public sector, and so criminals will be much more likely to use this encryption method to store their data.




(Mazarire, et al., 2016) Figure 3

What does all this mean for LEOs? Essentially, it is not easy to get the data criminals have. The government does have some tools, such as a much larger budget, at their disposal, so they are not at a total disadvantage. They have been arguing for several years that they need a "key under the mat" for encrypted systems, as well as asking for amendments to CALEA so that it can be useful in this new digital age.

### *Attacking Data at Rest*

The Going Dark problem was not really a huge issue in the 21st century until 2016 after the San Bernardino shooting. Tashfeen Malik and Syed Rizwan Farook killed 14 people and injured 22, but eventually perished in a firefight with police. Several people were arrested in connection with them, but the question the FBI struggled to answer was "Why?" (CNN, n.d.). The obvious way of solving this problem was to look at their phones. However, Farook owned an iPhone, and with the release of iOS 8 in September 2014, Apple encrypted all password protected information on their phones (Zittrain, 2016). This tied the key to the passcode that a user selected when setting up their phone, and all passcodes were stored locally on the phone itself. When the FBI asked Apple to unlock it, Apple's response was a polite "No thank you".

Obviously, the FBI was unhappy with this and so they sued Apple, leading to the first big modern-day conflict in the Going Dark problem.

The FBI was looking for a court order to compel Apple to assist them in unlocking Farook's iPhone. The central question in this case was could courts compel Apple (and presumably other tech companies) to release their customers' private, encrypted information to the government (Hennessey, 2016). Unfortunately, another company was able to unlock the phone using a zero-day exploit and so the government withdrew its motion. The bigger issue this case was pointing to, and one that many people would like to see settled, is does law enforcement have the right for "exceptional access", or a backdoor, to technology (Hennessey, 2016).

There are two ways the government can approach this problem. They can either continue to argue for exceptional access (like having keys under doormats) or they can plan to ignore the issue of exceptional access and instead jump right to legal hacking. Legal hacking may be more useful going after data in transit than data at rest, but that remains to be seen.

Two of the most popular scenarios that need to be thought through before a discussion of exceptional access can really begin are: first, "providing exceptional access to globally distributed, encrypted messaging applications" and second, "exceptional access to plaintext on encrypted devices such as smartphones" (Abelson et al., 2015). Some examples of encrypted messaging applications today are iMessage, Signal, and WhatsApp. The difference between the two scenarios is in the first, LEOs would be looking to gain access to the messages themselves, while in the second they would be seeking to gain access to other information on the phone that could be anything from emails to pictures.

Encrypted pieces of data have a key attached to them. Of course, depending on whether the encryption method was symmetric or asymmetric, this explains how the keys interact with each other. For the first scenario it would make sense to implement a symmetric key (Abelson et al., 2015). In order to give the government access to the global messaging system in the first scenario, it would make sense to encrypt the symmetric key a second time and use a special escrow key. This way, if LEOs obtains the data at rest (or in transit), they could use the escrow agent to decrypt the symmetric key which in turn would decrypt the data (Abelson et al., 2015).

There are three major issues with this idea. The first is purely geopolitical. Is the US able to reach into other countries encrypted messaging applications and grab information from there? The immediate answer is no, probably not but there are two more technical issues that would have to be settled first. One of the technical issues is that since the keys are symmetric, if a private key is ever leaked or breached, all data that has, is, or will ever be encrypted using that key is compromised (Abelson et al., 2015). The solution for that is something called forward secrecy, which is when long term keys are only used for authentication and a new key is created for each transaction. These session keys are destroyed once the transaction ends which cuts down on the data attackers can get it if a breach occurs. The other technical issue is that, by disclosing a key to a third party, even if it is the FBI, creates a whole other host of security vulnerabilities (Abelson et al., 2015).

The second scenario, in which LEOs would request, “exceptional access to plaintext on encrypted devices such as smartphones” (Abelson et al., 2015), has its own issues as well. It seems pretty straightforward though. Since several companies (not Apple) hold the keys needed to access the plaintext information on an encrypted device, it should be easy for a member of law enforcement to obtain a device, rip the identifying number off of it and send in a legally

substantiated request to a company in order to access the device's content. The issues with this scenario are a bit larger and have had a bit more real-world impact than issues with the first one. The biggest issue is one of authentication (Abelson et al. 2015). How can a request for a key be authenticated? How can the key itself be authenticated? How will this backdoor impact device performance? Ironically, one of the biggest examples of government hacking (Stuxnet) relied on taking advantage of company issued keys for encryption. Stuxnet was a virus developed by the National Security Agency (NSA) that they used to destroy centrifuges in Iran that were going to be used in their nuclear weapons program.

Both of these scenarios ignore the obvious problem: it simply does not benefit companies to give this data over to the government for free. Data has clearly begun to move into the forefront of many consumers' minds, and so they are quite happy with companies like Apple and Google giving their devices end to end encryption. Since those scenarios make exceptional access seem pretty unlikely, lawful hacking might provide a better way for LEOs to go after the data of criminals.

Lawful hacking is obviously a pretty controversial issue, because most people will not want the government to be snooping around their information, even if they have nothing to hide. Obviously, lawful hacking cannot be the magic bullet that solves the Going Dark problem. LEOs need to understand that there is simply some data that should and will remain encrypted beyond their reach. According to Susan Hennessey of the Brookings Institute, "lawful hacking should be viewed as the central element of a comprehensive alternative strategy" to combat the Going Dark problem (Hennessey, 2016). Hennessey saying that while lawful hacking will not solve the problem, it can certainly be used as a major part of the solution.

Lawful hacking does have quite a history of success in the US, and especially for the FBI. The FBI created a program called the Computer and Internal Protocol Address Verifier (CIPAV) that was a form of malware, and it would grab a lot of information from a computer where it was installed. Some of that information includes open ports, login names, and IP addresses. (Nguyen, 2017). CIPAV was able to take down a network of nearly 30,000 child predators on the dark web and also found the perpetrator of a bomb threat (Nguyen, 2017). Another example of lawful hacking is exploiting zero-day vulnerabilities in the interest of national security. The Stuxnet virus that was mentioned previously is actually the publicly released hack that the NSA performed on Iran prior to the Iran nuclear negotiations. Stuxnet was called "Olympic Games" when it was in-house with the NSA and was actually them taking advantage of several zero-day exploits to damage thousands of Iranian centrifuges.

For lawful hacking to be successful, it would have to be very carefully used and regulated. The way the FBI ran their investigation into the child predators, they first had to get a warrant to install CIPAV on a machine, then another warrant to be able to monitor communication on that machine (Nguyen, 2017). Some of these warrants were thrown out, because the warrant requires a specific location on them and if LEOs do not have an IP address there is no way for them to know exactly where the warrant will be executed. This oversight was fixed in 2016 when the US Supreme Court ruled that warrants on a computer did not need a specific geographic location to be executed (Nguyen, 2017). While that side of lawful hacking is cut and dry, the exploitation of zero-day vulnerabilities is not.

Does the government have a responsibility to businesses (and by extension, their customers) to report any vulnerabilities or flaws they find in software? The government's answer is a distinct "maybe." Both the software the FBI used to unlock the iPhone of the San Bernardino

shooters, as well as the vulnerability that led to the "Heartbleed" bug, were not revealed to the companies in charge of maintaining that software (Nguyen, 2017). The Vulnerability Equity Process (VEP) decides if a vulnerability gets reported or not. Of course, since they are dealing with issues of national security the process is rather opaque and so there is no real way to know the percentages of what gets reported versus what is not. The government claims approximately 91 percent are reported and 9 are not (Nguyen, 2017). Nguyen argues that since the average lifespan for a zero-day vulnerability is ten months, it does not matter if the government reports it or not since their investigation or operation should be completed by the time it is patched. However, there is some validity in arguing that the exact reasoning can be used to require that the government should report it as soon as they can. If they are going to exploit the vulnerability, it would make sense to show the company how it is occurring (and the process it took to find it) so the company can build a patch.

The government has several avenues available to them to attack data at rest. They can choose to go the route of requesting exceptional access, in which two scenarios are likely: they would need to be able to gain access to messaging applications that ignore arbitrary international borders, and they would have to navigate the security pitfalls that would come with exceptional access to plaintext on cell phones. If LEOs would like to avoid those legal issues, they could instead go the route of lawful hacking, which has already landed them in trouble thanks to the mishandling of zero-day exploits. However, there is a third route they could decide to go with that would require the help of Congress.

### *Amending CALEA*

Amending CALEA would allow the government to request exceptional access, or a "backdoor" into technology. Since CALEA was adopted in 1994, it is not equipped to handle the

modern-day technology of smart phones. It was most recently updated in 2006 when the FCC added broadband Internet and VoIP to the list of technology that falls under CALEA. However, this failed to cover internet messaging apps like Signal or Telegram (Nguyen, 2017). LEOs would like to see CALEA extended to text messaging, email and any other messaging apps that exist on the Internet, however implementing that for data in transit is a lot tougher than for data at rest.

There is not any specific legislation that has been introduced yet, however Manhattan District Attorney Cyrus Vance came up with his own set of guidelines he would like to see added. He essentially asks the developers of any operating system to ensure that they can access all information that is stored on a smartphone or tablet in an unencrypted format (Nguyen, 2017). While this proposal would be a good step, it only focuses on data at rest, and there is a whole other part of the going dark problem that it ignores.

While Vance's amendment fails to address data in transit, it solves the data at rest portion of the Going Dark problem. If this had been adopted, it would have made the San Bernadino shooter's iPhone a complete nonissue (Nguyen, 2017). The FBI would just have to, presumably, get a warrant for the shooter's iPhone and Apple would have had to produce it, unlocked and unencrypted. However, this is not the clean-cut solution the government makes it out to be. If there is exceptional access for the government (a backdoor), then it is almost guaranteed that other hackers would try to find this backdoor into iOS or any of the other mobile operating systems. While the government would need a warrant to access any encrypted information, any criminals out there would not. A possible solution to the worry of a backdoor is to only allow backdoor access to a whitelist of IPs (not very practical if it is to be used nationwide), or to only allow the backdoor to be unlocked with some sort of hardware component. Similar to how DoD

computers only work when a Common Access Card (CAC) is inserted, the backdoor could only be unlocked by a LEOs version of a CAC.

Amending CALEA is a really good way to address data at rest. Obviously, the amendment District Attorney Vance proposed has some issues, but it is a good place to start. There are very valid concerns about the ability of the developers working on the operating system to ensure the backdoor to provide exceptional access is not also a backdoor for possible criminals. They would obviously have to come up with some way to authenticate the user trying to access the phone, and that could be something like the LEOs version of a CAC card or even some sort of integration with the OS backdoor and the actual warrant they are trying to use. In order to go after data in transit, LEOs will have to take notes from the FBI and come up with more programs like Anom that are able to take advantage of the information that is not encrypted on apps like Signal or Telegram.

### *Data in Motion*

Data in motion is easier to retrieve for LEOs than data at rest, if only because they have been able to actually shift the platform being used. Generally, platforms like Signal and Telegram, or those built on the backs of XMPP (like the one the FBI built) have been locked down. In order to get to Anom (the FBI's messaging app), it is important to understand some of the other platforms that are used by people who do not want the eyes of big brother reading over their shoulder. Of course, Apple and Google already have established their end-to-end encryption, but that is not enough. Most people who are concerned about their security will use something like Signal or Telegram or more generically any app that utilizes the XMPP protocol, which is already utilized by WhatsApp and Zoom (XMPP, n.d.).

RSA is beginning to crack. While it is not cracking in the sense of being broken, it is beginning to crack portable computers (i.e., phones or tablets) with the processing power needed to create its keys (Kapoor et al., 2008). This is where Elliptical Curve Cryptography (ECC) comes in, and it is the cryptography method of choice for Signal and Telegram. ECC is based on a problem called the Elliptic Curve Discrete Logarithm problem and it is classified as NP-Hard (Kapoor et al., 2008). The classification of the problem is out of the scope of this paper, but it is enough to know that problems classified as NP-Hard are exactly that, hard. The reason elliptical curves are so helpful here is because they have a property that allows for a rule to be defined where two points on the curve can be added together to find a third point on the curve, and that is the basis of ECC. This third point is usually a n-bit prime that takes roughly 2^(n/2) operations to find (Kapoor, et al., 2008).

There are really three big advantages for ECC over RSA: security, space, and efficiency. In terms of security, the number of bits required for RSA to maintain the same level of security rises exponentially in comparison to ECC. For example, it takes 10^4 MIPS-years to break an RSA key with size 512 bits and an ECC key with size 106 (Kapoor et al., 2008). For a key that takes 10^8 MIPS years, an RSA key would be 768 bits while ECC would be only 132 (Kapoor et al., 2008). Clearly, since ECC requires less bits, this means that it also needs less transistors for the same level of security. For efficiency, an ECC 163-bit key takes 3.8 milliseconds to be generated while an RSA 1024-bit key (which equates to similar security levels) takes 4708.3 milliseconds (Kapoor et al., 2008).

Signal and Telegram both use their own specific versions of ECC that are based on the Diffie-Hellman functions and Signal breaks theirs down into four different ways to improve security. The first thing Signal does when encrypting is make a secure signature. They

accomplish this by using one of two algorithms: X Edwards-curve Signature Algorithm (XEdDSA) or Verifiable Random Function XEdDSA (VXEdDSA) (Perrin, 2016). The difference between those two algorithms is that VXEdDSA takes the existing XEdDSA and gives it an output that is guaranteed to be unique for both the message and the key (Perrin, 2016).

Another security feature Signal has is called X3DH, or Extended Triple Diffie-Hellman (Marlinspike, 2016). This is used as a key agreement protocol, and it provides that forward secrecy that was mentioned earlier, as well as cryptographic deniability. The point of X3DH is in case one user is offline, and another user wants to send data to them while also setting up a shared secret key for future communication (Marlinspike, 2016). It works like a lock box at a bank. Both users have keys, and they will leave messages in the lockbox for the other to retrieve and then respond. There are essentially 3 phases: one, where the first user will leave their identity and other relevant information in the lockbox, the second user fetches this information and uses it to send an initial message to the first user, and then the first user receives the message (Marlinspike, 2016). Since X3DH does not provide any authentication, the users will need something else (like a signature from XEdDsa) to authenticate that they are sending it to the correct user. However, there are some weaknesses with X3DH. Obviously, if the first phase key is compromised and it was not a one-time key, this could open up a world of opportunities for LEOs to either impersonate the user or string the other user along as an unknowing informant. Also, if the LEOs were able to compromise the server (lockbox) the users were operating on, that would enable LEOs to read all messages going back and forth, and even refuse to deliver messages.

Signal uses something called the Double Ratchet Algorithm to send messages, and when it is paired with X3DH key agreement it allows for an even more secure messaging protocol.

Double Ratchet works by creating new keys for every message so that if an earlier key is found the snooper can only use it to read one or two messages. They also use Diffie-Hellman for an extra layer of security (Marlinspike, 2016b). Sometimes messages can be sent out of order, but Double Ratchet deals with this by including the message number in the header, and this also helps solve the lost message problem. There are not too many ways for LEOs for attack this, however Double Ratchet allows its users to pick their own encryption algorithm to use for the messages and if that is broken obviously LEOs is helped.

The final part of Signal's messaging security is something called the Sesame Algorithm. It is designed to work with Double Ratchet messaging sessions that were created with X3DH key agreement, but it is generic enough that it functions with any message encryption that is session-based (Marlinspike, 2017). Sesame is designed based on the issues that arise from combining Double Ratchet and X3DH which include multiple devices being used and multiple sessions at the same time. There are several assumptions Sesame operates under, but they essentially all fall under there being some server they are communicating on, there are any number of users, and each user has at least one device at any time and they can add or remove devices at any time (Marlinspike, 2017). The basic way that Sesame works is that it defines the state that a device stores and then allows it to update its state while sending or receiving messages. One of the optional features is the ability to allow sessions to expire and new ones to be created for security purposes, which is obviously an issue for LEOs. One of the problems with having multiple devices is if one of them is compromised. This would be a big problem for LEOs. Of course, if a criminal is smart enough to set up Signal, they will also know to lock their phone behind a passcode so that would not be the huge help it seems to be at first.

While Signal has several different ways of protecting their client, Telegram uses their own encryption protocol called MTProto, and it is based on the AES encryption that was previously discussed. Telegram breaks their protocol down into two different kinds of chats: one for end-to-end encryption (secret chats) and a mobile protocol for chats over the cloud. Telegram also supports video chat but that just uses a modified version of the end-to-end encryption for secret chats. Immediately, it is pretty obvious that Signal is superior to Telegram for encryption, but it is still worth discussing Telegram to see different encryption options and ways they can be attacked.

The main difference for Telegram between secret chats and cloud chats is who holds the key. For secret chats, they have a key that is held only by the participants in the chat and that is regenerated periodically (End-to-End Encryption, Secret Chats, n.d.). They generate keys using the Diffie-Hellman protocol and then the actual chat process can begin. First, a request is created with some involved math (checking for primes and cyclic subgroups) and the request is then sent to another user. The second user will then accept the request and send back the authorization key that initiated the chat. This will allow the first user to compute the shared key and the chat can begin (End-to-End Encryption, Secret Chats, n.d.). In order to create forward secrecy, Telegram recommends recreating the key once it has been used for 100 messages or has been in use for a week, but that should just be general protocol for anyone trying to hide their messages.

MTProto for Telegram's cloud chat is built for mobile application to access a server API, but it does not work for web browsers (MTProto Mobile Protocol, n.d.). They break MTProto into three different components: a high-level component that converts API queries and response to "binary messages", a cryptographic component that encrypts messages before they are sent, and a transport component that transports the messages (MTProto Mobile Protocol, n.d.). That is

easily the least secure part of protocol, and so it is the part LEOs would target. Just like several other messaging protocols, the client and the server will use a session to exchange messages. This session will be attached to whatever the application is (whether it is the actual Telegram app or an app using Telegram's API) and several of these sessions can be open at once. In terms of encryption, Telegram is a bit lacking. They use AES-256, which is pretty strong but since it is a symmetric algorithm only one key is needed to break it, instead of the two needed for asymmetric. Finally for the transport component, they can use any of the protocols TCP, Websocket, Websocket over HTTPS, HTTP, HTTPS or UDP, but the problem with all of those is they all have different ways of attacking them. Obviously, since the message is encrypted, it only matters if LEOs have the key to decrypt them.

Telegram and Signal both offer different ways to protect the data and messages being sent by their users. Telegram uses AES-256 and MTProto to protect its users, while Signal uses four different kinds of algorithms and protocols that are based on elliptical curve cryptography. These apps present LEOs with the central problem of Going Dark. They know how to attack the data in transit, that is well documented, however once they get that data what is there to do? Well, one solution is to remove the middleman and actually become the messaging service that criminals will use.

### *Anom*

Anom is, so far, one of the biggest wins for law enforcement in the ongoing battle surrounding the Going Dark Problem. Anom was an app that law enforcement built to take the place of an app called Phantom Secure which was just another criminal encrypted messaging app. The app was based off of something called ArcaneOS, which is an Android operating system that heavily emphasizes privacy for people who value it (i.e., criminals). Anom was a

huge success for the FBI and other international partners, resulting in over 1,000 arrests worldwide (Cox, 2022).

Anom was built on the back of the XMPP messaging protocol, which is a very popular messaging standard. XMPP stands for Extensible Messaging and Presence Protocol, and it was originally built off of Jabber (An Overview of XMPP, n.d.). The reason they picked XMPP is because of its extensibility, which allows anyone to build custom functionality and use publicly available extensions. XMPP is built surrounding five key technologies: Core, Jingle, Multi-User Chat, PubSub, and BOSH. These five technologies allow XMPP to stream XML over a network, provide a way for Jabber users to manage their multimedia sessions, have multiple users in a singular "chatroom", a publishing and subscribing functionality, and a technology for two-way communication over HTTP, respectively (XMPP, n.d.).

Anom is a very simple application, but it created a blueprint for how LEOs can go about trying to solve the Going Dark problem. The basic premise of the app is that it is a messaging app that does not store any of the user's information. It was not too different from any of the other options on the criminal messaging market. The reason so many flocked to it is European and American authorities were going after two other popular messaging apps, Encrochat and Sky (Cox, 2022). Also, the FBI had a really good marketing strategy. They used criminal "influencers" to vouch for the app and make it seem like it was the next big thing (Baker et al., 2021). Someone in the market for an ANOM device could only get one if they knew the person selling them, so that gave it even more credibility. This process would come in handy later during the legal process as well.

The Anom phones came preloaded with an Anom help account, but also a hidden bot account. This hidden bot account is where all of the messages sent or received on the phone were

copied for LEOs to use later. This is a possible alternative to the backdoor that law enforcement has been seeking in technology. Instead of having to amend CALEA or deal with all the issues that would come with a policy change to exceptional access, instead law enforcement could obtain a warrant that would compel a messaging service to add a ghost account to the phone. This ghost account would only monitor sent messages, and it could be added either to a confiscated phone or remotely, literally exactly like how a wiretap works on phone lines today. It still allows for end-to-end encryption, however there is a third hidden end to that encryption. Not only did the app collect message information, but it also collected GPS coordinates of the phone and transferred it to the backend of the app (Cox, 2022).

Legally speaking, the FBI's decision to market the app through criminal word of mouth made ANOM one giant Racketeer Influenced and Corrupt Organizations (RICO) case (Baker et al., 2021). They essentially had set it up as something as simple as buying and using an ANOM phone made the user part of a criminal enterprise, which gave the FBI probable cause for searches or warrants. The FBI partnered with Australia's version of them (the AFP), and luckily enough Australia has a backdoor law on its books. This law, called the Australian Telecommunications and Other Legislation Amendment (Assistance and Access) Act basically allowed government agencies to issue notices that require communication service providers to basically make sure there are backdoors in their software for LEOs to use (Baker et al., 2021). The FBI never had any access to the messages that originated in America, and the AFP alerted them to any necessary threats to life. The cases are ongoing as of this writing, but it is likely this is the biggest win Law Enforcement has had in their attempts to solve Going Dark.

There are several possible solutions to the Going Dark Problem. Everything from legal hacking, to amending CALEA, to more solutions like ANOM. Of course, there is an argument

that the whole Going Dark problem is overblown. There is a stunning amount of information available online on social media. Any post made on social media has a ton of metadata available for LEOs to exploit. The recent advent of the Internet of Things also offers extensive opportunities for law enforcement. Of course, they would need to obtain warrants to access the information stored on an Alexa or a Samsung smart fridge, but that does not mean the information is not out there.

In order to solve Going Dark, law enforcement should attempt to come up with more solutions like ANOM but expand it to a criminal Twitter or Instagram; law enforcement should also try to move away from amending CALEA and towards legal hacking, since amending CALEA would create more problems. Law Enforcement is not going to be able to break RSA or AES or any of the ECC functions and so they should focus their energy on getting around the encryption rather than trying to find the keys to decryption. Creating apps like ANOM allows them to hold the keys to the castle, and that is far more powerful than any one decryption key.

Clearly, there are many different ways to attack the Going Dark Problem. However, to attack it, first it has to be understood. That begins with a discussion of data at rest, attacking it, and ways to amend CALEA. After that, it moves to a discussion of data in motion, attacking data in motion and Anom. Finally different ways to solve the Going Dark Problem were discussed.

Works Cited

Cox, J. (2022, July 7). *This Is the Code the FBI Used to Wiretap the World*. Retrieved September 11, 2022, from https://www.vice.com/en/article/v7veg8/anom-app-source-code-operation-trojan-shield-an0m

Cox, J. (2022a, January 4). *FBI's Backdoored Anom Phones Secretly Harvested GPS Data Around the World*. Retrieved September 11, 2022, from https://www.vice.com/en/article/93b3ay/fbi-backdoor-anom-phones-gps-data

@BryceKlehm. (2021, October 16). *Legal Tetris and the FBI's ANOM Program*. Lawfare. Retrieved September 11, 2022, from https://www.lawfareblog.com/legal-tetris-and-fbis-anom-program

Farzan, & Taylor. (2021, June 8). *What is Anom, and how did law enforcement use it to arrest hundreds in a global sting?* Washington Post. Retrieved September 11, 2022, from https://www.washingtonpost.com/world/2021/06/08/an0m-sting-faq/

*XMPP | An Overview of XMPP*. (n.d.). Retrieved September 11, 2022, from https://xmpp.org/about/technology-overview/

Comey. (n.d.). *Going Dark: Are Technology, Privacy, and Public Safety on a Collision Course?* https://www.fbi.gov/news/speeches/going-dark-are-technology-privacy-and-public-safety-on-a-collision-course

Traylor, J. (2017). Shedding Light on the "Going Dark" Problem and the Encryption Debate. *University of Michigan Journal of Law Reform, 50.2*, 489. https://doi.org/10.36646/mjlr.50.2.shedding

Hennessey, S. (2022, March 9). *Lawful hacking and the case for a strategic approach to "Going Dark."* Brookings. Retrieved September 11, 2022, from https://www.brookings.edu/research/lawful-hacking-and-the-case-for-a-strategic-approach-to-going-dark/

Nguyen. (n.d.). *LAWFUL HACKING: TOWARD A MIDDLE-GROUND SOLUTION TO THE GOING DARK PROBLEM* [Master's Thesis]. Naval Postgraduate School.

Walden, I. (2018, August). 'The Sky is Falling!' – Responses to the 'Going Dark' problem. *Computer Law &Amp; Security Review*, *34*(4), 901–907. https://doi.org/10.1016/j.clsr.2018.05.013

Zittrain, Jonathan L., Matthew G. Olsen, David O'Brien, and Bruce Schneier. 2016. "Don't Panic: Making Progress on the "Going Dark" Debate." Berkman Center Research Publication 2016-1.

Chen, C. W. (2017). The graymail problem anew in a world going dark: Balancing the interests of the government and defendants in prosecutions using network investigative techniques (NITs). *Colum. Sci. & Tech. L. Rev.*, *19*, 185.

Manpearl, E. (2017). Preventing Going Dark: A Sober Analysis and Reasonable Solution to Preserve Security in the Encryption Debate. *U. Fla. JL & Pub. Pol'y*, *28*, 65.

Koops, B. J., & Kosta, E. (2018). Looking for some light through the lens of "cryptowar" history: Policy options for law enforcement authorities against "going dark". *Computer Law & Security Review*, *34*(4), 890-900.

Podhradsky, A., D'Ovidio, R., & Casey, C. (2012). The XBOX 360 and Steganography: How Criminals and Terrorists Could Be" Going Dark".

Christie, James, "“Going Dark” – The Challenge Facing Law Enforcement in the 21st Century" (2019). Economic Crime Forensics Capstones. 45. https://digitalcommons.lasalle.edu/ecf_capstones/45

Blaze, M. (1994). Protocol failure in the escrowed encryption standard. *Proceedings of the 2nd ACM Conference on Computer and Communications Security - CCS ’94*. https://doi.org/10.1145/191177.191193

Bernstein v. U.S. Dep't of Justice, 176 F.3d 1132, 1145-46 (9th Cir. 1999), reh'g granted, opinion withdrawn, 192 F.3d 1308 (9th Cir. 1999).

Daoud, Luka; Hussein, Fady; and Rafla, Nader. (2019). "Optimization of Advanced Encryption Standard (AES) Using Vivado High Level Synthesis (HLS)". Proceedings of 34th International Conference on Computers and Their Applications, 58, 36-44.

Daoud, L., & Huen, H. (n.d.). Performance Study of Software-based Encrypting Data at Rest. *EPiC Series in Computing*. https://doi.org/10.29007/1j1p

*The DES Algorithm Illustrated*. (n.d.). Page.math.tu-Berlin.de. https://page.math.tu-berlin.de/~kant/teaching/hess/krypto-ws2006/des.htm

Fatima, S.; Rehman, T.; Fatima, M.; Khan, S.; Ali, M.A. Comparative Analysis of Aes and Rsa Algorithms for Data Security in Cloud Computing. Eng. Proc. 2022, 20, 14. https://doi.org/10.3390/ engproc2022020014

CNN, S. A. (n.d.). *Who were Syed Rizwan Farook and Tashfeen Malik?* CNN. https://www.cnn.com/2015/12/03/us/syed-farook-tashfeen-malik-mass-shooting-profile/index.html

Abelson, H., Anderson, R., Bellovin, S. M., Benaloh, J., Blaze, M., Diffie, W., Gilmore, J., Green, M., Landau, S., Neumann, P. G., Rivest, R. L., Schiller, J. I., Schneier, B., Specter, M. A., & Weitzner, D. J. (2015). Keys under doormats: mandating insecurity by requiring government access to all data and communications. *Journal of Cybersecurity*, *1*(1), 69–79. https://doi.org/10.1093/cybsec/tyv009

Kapoor, V., Abraham, V. S., & Singh, R. (2008). Elliptic curve cryptography. *Ubiquity*, *2008*(May), 1–8. https://doi.org/10.1145/1386853.1378356

Perrin, T. (Ed.). (2016, October 20). *Specifications >> The XEdDSA and VXEdDSA Signature Schemes*. Signal Messenger. https://signal.org/docs/specifications/xeddsa/#ref-ed25519

Marlinspike, M. (2016, November 4). *Signal Messenger: Privacy That Fits in Your Pocket* (T. Perrin, Ed.). Signal Messenger. https://signal.org/docs/specifications/x3dh/

Marlinspike, M. (2016b, November 20). *Signal Messenger: Speak Freely* (T. Perrin, Ed.). Signal Messenger. https://signal.org/docs/specifications/doubleratchet/

Marlinspike, M. (2017, April 14). *Specifications >> The Sesame Algorithm: Session Management for Asynchronous Message Encryption* (T. Perrin, Ed.). Signal Messenger. https://signal.org/docs/specifications/sesame/

*End-to-End Encryption, Secret Chats*. (n.d.). Core.telegram.org. https://core.telegram.org/api/end-to-end

*MTProto Mobile Protocol*. (n.d.). Core.telegram.org. https://core.telegram.org/mtproto

McLoone, M., & McCanny, J. V. (2003). High-performance FPGA implementation of DES using a novel method for implementing the key schedule. *IEE Proceedings - Circuits, Devices and Systems*, *150*(5), 373. https://doi.org/10.1049/ip-cds:20030574

Mazarire, Jothina & Kwenda, Clopas. (2016). Hybrid Algorithm for E-commerce Applications. International Journal of Research in IT & Management. 5.