The Idiot Savants' Guide to Rubberhose

Suelette Dreyfus

--- The third name is MARUTUKKU, Master of the arts of protection, chained the Mad God at the Battle. Sealed the Ancient Ones in their Caves, behind the Gates.

The Akkadian Creation Epic

1. What is Rubberhose?

Rubberhose is a computer program which both transparently encrypts data on a storage device, such as a hard drive, and allows you to hide that encrypted data. Unlike conventional disk encryption systems, *Rubberhose* is the first successful, freely available, practical program of deniable cryptography in the world. It was released in an earlier form in 1997, but has undergone significant changes since that time. The design goal has been to make Rubberhose the most efficient conventional disk encryption system, while also offering the new feature of information hiding.

Rubberhose is a type of deniable cryptography package. Deniable cryptography gives a person not wanting to disclose the plaintext data corresponding to their encrypted material the ability to show that there is more than one interpretation of the encrypted data. What deniable crypto means in the Rubberhose context is this: if someone grabs your Rubberhose-encrypted hard drive, he or she will know there is encrypted material on it, but not how much -- thus allowing you to hide the existence of some of your data.

Written in C, Rubberhose supports NetBSD and Linux as a suite of kernel modules and userland programs. The program has worked under FreeBSD in the past, but the

FreeBSD port needs updating to reflect the new kernel-userland messaging system. (Hands up, anyone who wants to join our team for this?) We are planning to port it to Windows NT™ as well -- and are looking for volunteers here as well.

Rubberhose was originally conceived by crypto-programmer Julian Assange as a tool for human rights workers who needed to protect sensitive data in the field, particularly lists of activists and details of incidents of abuse. Repressive regimes in places like East Timor, Russia, Kosovo, Guatamalia, Iraq, Sudan and The Congo conduct human rights abuses regularly. Our team has met with human rights groups an heard first hand accounts of such abuses. Human rights workers carry vital data on laptops through the most dangerous situations, sometimes being stopped by military patrols who would have no hesitation in torturing a suspect until he or she revealed a passphrase to unlock the data. We want to help these sorts of campaigners, particularly the brave people in the field who risk so much to smuggle data about the abuses out to the rest of the world.

Of course, you don't have to be a human rights activist to play with Rubberhose -- we invite everyone from tinkerer coder to nefarious cypherpunk to use this code. Those coders from European and North American countries who test Rubberhose may soon find their efforts pay off closer to home. Draconian new crypto laws currently proposed in Britain call for a prison term of up to two years for anyone who refuses to turn over his or her encryption key, or the messages in plaintext, to law enforcement agents. If passed, the Electronic Communications Bill will also see anyone tipping off a target about an investigation looking forward to up to five years in prison. In fact, under the new law, even complaining about alleged abuses by law enforcement to the media may land whistle-blowers a prison term. Other Western governments have are to a greater or lesser degress following Britian's lead.

If you're wondering about the name of this program, Marutukku is the internal development name (it's spelled R-u-b-b-e-r-h-o-s-e, but it's pronounced M-a-r-u-t-u-k-k-u) and Master of the arts of protection, as described in ancient Mesopotamian creation work *The Akkadian Creation Epic*. Read all seven tablets (a translation of the tablets is contained in the file ENUMAELISH (http://www.rubberhose.org/current/src/ENUMAELISH). Or you could try the MYTHOLOGY (http://www.rubberhose.org/current/src/MYTHOLOGY) file distributed with the program instead. We hope that Rubberhose will both protect your data, and offer a broader kind of protection for people who take risks for just causes.

Civilization advances slowly, and traditionally it is human rights campaigners, free-thinking academics and community activists who propel it forward. Society often resists these gingerly-placed prods, and the entrenched moguls punish those who dare to upset the status quo. They label the activists as trouble-makers or whistle blowers to justify misusing them. Where there is injustice, we like to upset the status quo too, and to support others who want to do the same. Our motto is "let's make a little trouble." The Rubberhose development team is releasing Rubberhose to help protect activists who want to make a little trouble. Feel free to make a little trouble yourself.

2. What can you do with Rubberhose?

You can hide information effectively and safely.

Here's a simple example of Rubberhose's usefulness. Rashid, a civil rights community activist, has three types of data on his computer: his bank statements, a list of witnesses statements about a government-instigated death squad attack on a rural village, and his mother's recipe for rhubarb tart. The tart recipe, and, to a lesser degree, his banks statements, are his cover. They provide decoy data for the more explosive witness statements underneath.

In the middle of the night, military security forces raid Rashid's home, take him prisoner and seize his computer. Computer experts at the Ministry of Cleansing of Public Ideas examine his machine and find that the hard drive is encrypted. They demand the passphrase in less than subtle ways in order to decrypt the data. Rashid gives them a passphrase. They decrypt the data and discover a recipe for rhubarb tart. They poke around the drive, but they can not see any other encrypted data, because there is no tractable way to show the existence of any other data hidden among the rhubarb. They are frustrated and angry but can trump up no suitable charges to hold Rashid. There is no way for the interrogators -- or Rashid himself -- to prove he has handed over all the passphrases. Finally releasing him and his computer, the interrogators never know the bank statements or witness statements are on the machine.

Rubberhose also allows you to pass information around to a group of people, but only allow certain people to access certain material. Rashid might, for example, discover his

colleague Leila is on a military hit list and need to smuggle her out of the country to safety. To do this, she will need to pass through four or five safe-houses run by secret sympathizers on the way to the border. Rashid knows all the safe houses, but he does not want them to know about each other for fear that if the military regime seized one person, the other four will be at risk. So Rashid gives Leila a floppy disk with the list of safe-houses -- but each name is encrypted with a different passphrase. Each secret sympathizer knows his or her own passphrase. When Leila arrives at a safe house, she gives the diskette to the sympathizer, who decrypts his portion which contains the address of the next safe house in the chain. Leila never knows the entire list of safe house sympathizers until she is safely over the border; if she is ever intercepted en route with the diskette the list of names is safe. Other crucial information can also be passed along a human chain in this way.

The multi-passphrase security feature which allows information hiding only works for reading data. You can only safely write to the diskette or other data storage device with Rubberhose if you know all the passphrases. If you try to write to the disk without first decrypting the entire disk, you risk overwriting information stored in some other encrypted portion. If however if it more important to keep the information hidden than it is to keep it intact, this shouldn't bother you. Rubberhose will allow you to over-write the material. Naturally the program will not warn you that you are doing so, however, as this might tip off an adversary that there is something far more valuable than just a tart hiding in the Rubberhose disk.

3. How does Rubberhose work?

3.1. The Broad Overview

When you run Rubberhose over a disk for the first time, the program writes random characters to the entire drive. (This means you should not run Rubberhose over a hard drive or any other storage device which is the sole repository of your Swiss bank account numbers.) The most important feature of Rubberhose is that this random noise

generated in the initialization is indecipherable from the encrypted data which will eventually be stored on the disk.

Let's say the hard drive is 1GB. You want to fill the first Rubberhose-encrypted portion with 400MB. and the second with 200MB. When you do this, Rubberhose doesn't know that you intend to divide up the sections in this way: the program assumes each of the sections will be 1GB, and therefore will fill the whole drive. In practice, Rubberhose fills the drive on a first come, first serve basis: it will keep providing more room to any one section until the overall disk has reached capacity.

However, when Rubberhose is creating these "partitions" it doesn't simply cordon off the left corner of the drive. It doesn't work like a normal disk partitioning program, taking large blocks for each partition. Instead, it breaks up the pieces of the 400MB encrypted portion into tiny pieces and scatters them across the entire 1GB drive. This is done in a random manner, so the bits of data can not be tracked and re-assembled by an adversary. When you decrypt that 400MB section, it will look as though it is actually 1GB in size, with 600MB free space. This structure is how Rubberhose hides the existence of data in the remaining portion of the disk.

Now, you make a second Rubberhose encrypted portion of 200MB. Again, the program writes these bits randomly all over the 1GB drive as well, while taking care to avoid over-writing the bits assigned to the first 400MB portion.

Since Rubberhose strews the bits all over the drive, we decided to call a Rubberhose section an "aspect", rather than a partition or level. Each Rubberhose aspect has its own passphrase and must be decrypted separately. The entire Rubberhose drive is an "extent". The idea behind choosing the word aspect is simple: a Rubberhose "aspect" is simply a view of the same thing (the extent) from a different perspective -- like viewing the same object from different angles. The program lets you make as many aspects in one extent as you want. If anyone seizes your data, they will not be able to tell either through mathematical analysis or physical testing of the disk how many aspects are on the drive, or how much real data is on any one aspect.

Rubberhose relies on internal maps to locate where the actual bits of your data are stored amid the random characters. Each aspect has its own corresponding map, and you can only decrypt that aspect's map when you type in the passphrase for that particular aspect. This is why you need to type in all the passphrases before you can

safely write to a Rubberhose disk.

Like the many well disciplined spy agencies which will no doubt try to pick apart this program as soon as it is released, Rubberhose works on a strictly "need to know" basis. A Rubberhose aspect "knows" nothing about any other aspect -- including its size, maps or even existence. The program is designed this way because an aspect that doesn't know, can't tell anyone else. The only thing it needs to know is when it must avoid writing over the top of another aspect.

As a security measure, Rubberhose doesn't actually give any one aspects a piece of real estate on the disk until you actually try to write to the disk. (In other words, parts of the extent are only doled out to the various aspects dynamically.) This instant generation reduces the risk of analysis by an enemy, and allows the information hiding. For reading any one aspect, Rubberhose doesn't need to know the maps of the other aspects. For read only, when you type in one passphrase, Rubberhose will only decrypt the tables which map the bits for that one single aspect; all the other mapping tables for other Rubberhose aspects stay securely encrypted.

3.2. Cryptographic Algorithms You Can Use in Rubberhose

We like our privacy. So Rubberhose uses only the best cryptographic algorithms available. You can set up your version of Rubberhose to use any of the following, alone or in combination:

DES 3DES IDEA RC5 RC6 Blowfish Twofish CAST

* *RC2 can be used as well, however this is not recommended.*

The deniable aspects of Rubberhose are built on top of the interaction of strong cryptography algorithms. The protection of the data itself, regardless of the deniable aspects, also relies on these strong, peer-reviewed algorithms.

In addition to the built in ciphers, Rubberhose supports all the symmetric algorithms

from the latest release of OpenSSL, the world's fastest and most well-known crypto-library. Rubberhose's modular design makes it very easy to add new algorithms as well.

For the legal Riverdance, please read the PATENTS (http://www.rubberhose.org/current/src/PATENTS) file in the distribution.

3.3. File Systems and Rubberhose

The good news is that Rubberhose is very flexible: you can run it with a wide range of file systems, such as UFS, ext2fs, FAT or FAT32. Or any file-system your operating system uses to format a disk. However, we *strongly* recommend that you do not run a log-structured file system with Rubberhose. To understand why, read Section 4.3.

4. Specific Security Features

4.1. Time-Based Passphrases

Rubberhose allows you to set time-based passphrases. For example, you can set up Rubberhose to demand re-entry of a passphrase after one hour of use, or any other time length you prefer. This is a simple but crucial safety feature. If your as yet unpaid lawyer burst through your front door and ties you up 50 minutes into the hour, he will only have 10 minutes to extract all your Cayman Island account numbers from your drive before Rubberhose demands the passphrase -- and then cuts him off.

Rubberhose can also be configured to lock out users after a certain amount of idle time. This security feature means that if you stroll away from your machine to nibble on some rhubarb tart, and one slice happens to turn into an extended feeding fest (as always happens with rhubarb tart), your valuable data will not be exposed to attack for very long.

4.2. Anti-Passphrase-Cracking Features

Rubberhose is highly resistant to dictionary attacks. Every time you create a new aspect, the program generates an internal master password. You never see this password, which is just as well, since you probably couldn't remember it anyway. The aspect's internal master password is very long and randomly generated (and therefore extremely difficult to guess). Typically, it includes the kind of obscure characters less experienced computer users only stumble across by accident (ever mistyped and wondered what that yen symbol was doing on your screen?) Rubberhose then encrypts this internal master passphrase with the passphrase you give the program for that particular aspect.

The internal master password stays the same for the life of the aspect, but you can change your passphrase for that aspect as many times as you like.

> **Technical Guff:** By default, when you create a new Rubberhose passphrase, the program loops for a period of time re-encrypting the first output into the second, and the second into the third, etc. Each subsequent encryption output is fed back into the next round of encryption. This makes an attack based on guessing the passphrase extremely difficult, as an attacker must test each guess through the entire loop.

You can set the looping time to your own preference in Rubberhose each time you first open the program. Longer times mean it will take slightly longer to decrypt an aspect every time you open it, but the data will be more secure. For example, a two second loop time setting run on an Intel Celeron™ 366 means Rubberhose will do 175,000 rounds of encryption (for the CAST cipher). The only way for an attacker to return to the original master password is to reverse the process, feeding the output of the decryption into the input of the next decryption -- through about 175,000 rounds, depending on the speed of the symmetric cipher algorithm selected.

The salted internal master aspect key unlocks everything in that aspect, including the map of where the bits of data are located across the drive and the ability to actually decrypt that data. Because this key could be a point of vulnerability, we have designed large walls around the master key. In fact, it could be said that the master key doesn't

touch anything else in the program directly; there is always an obscuring barrier protecting it.

4.3. Thwarting Disk Surface Analysis

There are several well-known attacks on encrypted data via surface analysis of the disk itself. Rubberhose is designed to protect against these attacks, including attacks based on scanning tunnelling microscopy advances.

The program also ensures that if an attacker manages to decrypt one of the many tiny blocks of data in an aspect, it will not help him or her to decrypt the remaining blocks and therefore piece together all the data from an entire aspect. Our team reads academic papers on cryptanalytic attacks for fun on the weekends. The Rubberhose team has tried to make this program reasonably immune from the latest attacks of this type.

> **Technical Guff:** Using strong pseudo-random number generators, Rubberhose creates a lattice generator for each aspect. This lattice, which can only be decrypted by using the internal master key for that aspect, is used for calculating unique encryption and decryption keys for each block of data assigned to a single aspect. Specifically, the lattice uses a mathematical algorithm to transmute a block number (let's say, Block Number 42, as it holds the answer to a particularly interesting question) into a key for that block. Rubberhose puts these blocks inside its own larger blocks.

There are two types of blocks in a Rubberhose-encrypted drive: an operating system block (typically 512-8192 bytes) and a Rubberhose "surface-block" (which is usually congruent with an OS block size but not always), which are scattered all over the drive in a truly random fashion. You can configure the size of the surface blocks to suit your needs when you initialize Rubberhose on your hard drive. Setting Rubberhose to create many smaller surface blocks is potentially a little more secure than choosing larger block sizes, but the blocks also take longer to generate.

The program is implemented in such a way that breaking one block of data will not aid an attacker in breaking the next one. More importantly, it is very difficult to discover

statistical inter-relationships between large numbers of blocks, even if the block cipher algorithm is weak and the whitener (described later) is broken.

Each aspect is in fact a "view", or map, of these blocks -- but when you examine Aspect 1, the Aspect appears to view *all* of the disk as belonging to it. Similarly, Aspect 2 views all of the block as its own. An attacker examining Aspect 1 would simply see some used blocks and some unused blocks within the single aspect. In fact, some of those "unused" blocks might secretly belong to Aspect 2, but Aspect 1 can not tell this, so neither does the attacker who has accessed Aspect 1.

Rubberhose conducts frequent block-swapping to thwart disk surface analysis based on the intensity of block use and contiguous block prediction. In theory an attacker can examine the magnetic properties of the ferrite coating on a disk surface in order to determine how frequently a program has read or written to a particular section of the drive. This permits the attacker to guess if a geographic area on the disk is blank (full of random noise) or contains hidden data. If the attacker can decrypt, for example, Aspect 1 (but not any other Aspect) he can overlay a map of frequently used drive sections on a map of Aspect 1's data map showing unused and used sections. If he sees an unused section has been accessed for reading or writing very frequently, he can guess that there is more likely hood than not that there is hidden material stored there from another aspect.

To prevent this sort of analysis, Rubberhose automatically shuffles surface blocks of data around to new locations on the drive with a frequency you can set. It does this regularly but invisibly, so the swapping doesn't interfere with your work in progress. The background shuffling occurs whenever Rubberhose is open (with write capability), even when your machine is idle. The result is that no one region of the disk looks more used than any other section in a statistically significant fashion.

Further, this block shuffling reduces the risk of contiguous block analysis. If an attacker decrypts Aspect 1 and finds all the data written in a contiguous block on the drive, he could reasonably guess the block immediately next to it contains hidden data written from Aspect 2. By moving its blocks around, Rubberhose avoids the predictability of tidy, contiguous block assignment. Take that, Ahab.

4.4. Cryptanalytic Attacks (Known Plaintext Attacks etc.)

Rubberhose also contains "whitening code", another security feature designed to thwart cryptanalysis attacks, particularly "Known Plaintext attacks". The whitener is made out of random bits. Whitener also perturbs the encryption process in a non-predictable manner.

Rubberhose merges this whitener and decrypted data from the data blocks in order to change the data in the data blocks slightly. Specifically, the whitener "flips" a corresponding bit in the data block. Any bit in the whitener which is a 1 flips a corresponding bit in the decrypted data block. Any 0 means "hold", or "don't flip". The result is that some bits in the decrypted data block are flipped and some are not. This prevents known plaintext attacks because you don't have any plaintext to work with which is directly translatable into the encrypted material.

Whitener is vulnerable in the sense that if you break it for one block you will probably break it for all the blocks. However, there are still other lines of defense, such as separate keys for every block which limit the success of such an attack.

5. Licensing

Rubberhose costs nothing, unless you are a non-humanitarian government organisation or commercial entity. You may run Rubberhose for your own personal use, human rights or community group activities or for non-defense related academic research.

If you want to use the program in a commercial setting, you need to seek a license. If you work for a state security agency, spy agency, military or defense organisation you must also seek a license before using this program.

Any proceeds go to the Melbourne Institute for Advanced Study a non-profit research organisation.