Snake Oil Warning Signs: Encryption Software to Avoid

Matt Curtin <cmcurtin@interhack.net>

April 10, 1998

Contents

Administrativia

Distribution

Distribution of this document is unlimited. We're specifically trying to reach people who are not experts in cryptography or security but find themselves making decisions about what sorts of crypto (if any) to use, both for their organizations and for themselves.

The Snake Oil FAQ is posted monthly to `sci.crypt`, `alt.security`, `comp.security`, `comp.answers`, and `comp.infosystems`. It is available in PostScript and PDF form (ideal for printing) via the web at

```
http://www.interhack.net/people/cmcurtin/snake-oil-faq.ps
http://www.interhack.net/people/cmcurtin/snake-oil-faq.pdf
```

and HTML at

```
http://www.interhack.net/people/cmcurtin/snake-oil-faq.html
```

Disclaimer

All contributors' employers will no doubt disown any statements herein. We're not speaking for anyone but ourselves.

This is a compilation of common habits of snake oil vendors. It cannot be the sole method of rating a security product, since there can be exceptions to most of these rules. From time to time, a reputable vendor will produce something that is actually quite good, but will promote it with braindead marketing techniques. But if you're looking at something that exhibits several warning signs, you're *probably* dealing with snake oil.

Every effort has been made to produce an accurate and useful document, but the information herein is completely without warranty. This is a work-in-progress; feedback is greatly appreciated. If you find any errors or otherwise wish to contribute, please contact the document keeper, Matt Curtin <cmcurtin@interhack.net>

Document History

With the rise in the number of crypto products came a rise in the number of ineffective or outright bogus products. After some discussion about this on the cypherpunks list, Robert Rothenburg <wlkngowl@unix.asb.com> wrote the first iteration of the Snake Oil FAQ. Matt Curtin took the early text and munged it into its current state with the help of the listed contributors (and probably some others whose names have inadvertently missed. Sorry in advance, if this is the case.)

Contributors

The following folks have contributed to this FAQ.
Jeremey Barrett <jeremey@forequest.com>
Steven M. Bellovin <smb@research.att.com>
Matt Blaze <mab@research.att.com>
Bo Dömstedt <bo.domstedt@protego.se>
Gary Ellison <gary.ellison@eng.sun.com>
<fifersl@ibm.net>
<geeman@best.com>
Larry Kilgallen <KILGALLEN@Eisner.DECUS.Org>
Dutra Lacerda <dutra.lacerda@mail.telepac.pt>
Felix Lee <flee@teleport.com>
Colin Plumb <colin@nyx.net>
Jim Ray <jmr@shopmiami.com>
Terry Ritter <ritter@io.com>
Robert Rothenburg <wlkngowl@unix.asb.com>
Adam Shostack <adam@homeport.org>
Rick Smith <smith@sctc.com>
Randall Williams <ac387@yfn.ysu.edu>

1 Introduction

Good cryptography is an excellent and necessary tool for almost anyone. Many good cryptographic products are available commercially, as shareware, or free. However, there are also extremely bad cryptographic products which not only fail to provide security, but also contribute to the many misconceptions and misunderstandings surrounding cryptography and security.

Why "snake oil"? The term is used in many fields to denote something sold without consideration of its quality or its ability to fulfill its vendor's claims. This term originally applied to elixirs sold in traveling medicine shows. The salesmen would claim their elixir would cure just about any ailment that a potential customer could have. Listening to the claims made by some crypto vendors, "snake oil" is a surprisingly apt name.

Superficially, it is difficult to distinguish snake oil from the Real Thing: all encryption utilities produce garbled output. The purpose of this document is to present some simple "red flags" that can help you detect snake oil.

For a variety of reasons, this document does not mention specific products or algorithms as being "good" or "snake oil."

2 Basic Concepts

In an effort to make this FAQ more complete, some basic information is covered here. The Cryptography FAQ [3] is a more general tutorial of cryptography and should also be consulted.

When evaluating any product, be sure to understand your needs. For data security products, what are you trying to protect? Do you want a data archiver, an e-mail plug-in, or something that encrypts on-line communications? Do you need to encrypt an entire disk or just a few files?

And how secure is secure enough? Does the data need to be unreadable by "spies" for five minutes, one year, or 100 years? Is the spy someone's kid sister, a corporation, or a government?

2.1 Symmetric vs. Asymmetric Cryptography

There are two basic types of cryptosystems: symmetric (also known as "conventional" or "secret key") and asymmetric ("public key.")

Symmetric ciphers require both the sender and the recipient to have the same key. This key is used by the sender to encrypt the data, and again by the recipient to decrypt the data. The problem here is getting the sender and recipient to share the key.

Asymmetric ciphers are much more flexible from a key management perspective. Each user has a pair of keys: a public key and a private key. Messages encrypted with one key can only be decrypted by the other key. The public key can be published widely while the private key is kept secret.

So if Alice wishes to send Bob some secrets, she simply finds and verifies Bob's *public* key, encrypts her message with it, and mails it off to Bob. When Bob gets the message, he uses his *private* key to decrypt it.

Verification of public keys is an important step. Failure to verify that the public key really does belong to Bob leaves open the possibility that Alice is using a key whose associated private key is in the hands of an enemy.

Asymmetric ciphers are much slower than their symmetric counterparts. Also, key sizes generally must be much larger. See the Cryptography FAQ [3] for a more detailed discussion of these topics.

2.2 Secrecy vs. Integrity: What are you trying to protect?

For many users of computer-based crypto, preserving the contents of a message is as important as protecting its secrecy. Damage caused by tampering can often be worse than damage caused by disclosure. For example, it may be disquieting to discover that a hacker has read the contents of your funds-transfer authorization, but it's a disaster for him to change the transfer destination to his own account.

Encryption by itself does not protect a message from tampering. In fact, there are several techniques for changing the contents of an encrypted message without ever figuring out the encryption key. If the integrity

Table 1: Time and Cost of Key Recovery

Type of Attacker	*Budget*	*Tool*	*Time and Cost per 40-bit Key Recovered*	*Length Needed for Protection in late 1995*
Pedestrian Hacker	Tiny	Scavenged Computer Time	1 Week	45
	$400	FPGA	5 Hours ($0.08)	50
Small Business	$10,000	FPGA	12 Minutes ($0.08)	55
Corporate Department	$300K	FPGA	24 seconds ($0.08)	60
		ASIC	.005 seconds ($.001)	
Big Company	$10M	FPGA	.7 seconds ($0.08)	70
		ASIC	.0005 seconds ($0.001)	
Intelligence Agency	$300M	ASIC	.0002 seconds ($0.001)	75

of your messages is important, don't rely on just secrecy to protect them. Check how the vendor protects messages from undetected modification.

2.3 Key Sizes

Even if a cipher is secure against analytical attacks, it will be vulnerable to brute-force attacks if the key is too small. In a brute-force attack, the attacker simply tries every possible key until the right one is found. How long this takes depends on the size of the key and the amount of processing power available. So when trying to secure data, you need to consider how long it must remain secure and how much computing power an attacker can use.

[1] and [2] offer some guidelines for choosing an appropriate key length. For instance, Table 1 shows the cost of breaking symmetric keys by brute force, as noted by [2]. This same report strongly recommends using symmetric keys of 90 bits or more.

With the tremendous increases in computing power over the last several decades, cryptosystems which were once considered secure are now vulnerable to brute-force attacks. RSA Laboratories sponsored a series of contests, collectively known as the *1997 Secret Key Challenge* [5]. So far, we have seen RC5 up to 56 bits fall victim to brute force attacks, as well as the financial industry's workhorse, DES. At 56 bits, the keys used for DES are just too small to stand up to a dedicated attacker. It's noteworthy that both of the groups to break a DES-encrypted message did so with essentially no funding.

As mentioned earlier, asymmetric ciphers typically require significantly longer keys to provide the same level of security as symmetric ciphers. Comparing key lengths between algorithms is awkward because different algorithms have different characteristics. Knowing the key size is useless if you don't know what type of algorithm is being used.

But to give you some idea of what's reasonable, Table 2, from [1], compares symmetric keys against one type of asymmetric key: those based on the "factoring problem" or the "discrete log problem." (Algorithms based on the "elliptical curve discrete log problem" are more resistant to brute-force attacks and can use much smaller keys. In fact, they don't have to be much larger than symmetric keys, as far as we know right now.)

Table 2: Key Lengths With Similar Resistance to Brute-Force Attacks

Symmetric Key Length	*Public Key Length*
56 bits	384 bits
64 bits	512 bits
80 bits	768 bits
112 bits	1792 bits
128 bits	2304 bits

2.4 Keys vs. Passphrases

A "key" is not the same thing as a "passphrase" or "password." In order to resist attack, all possible keys must be equally probable. If some keys are more likely to be used than others, then an attacker can use this information to reduce the work needed to break the cipher.

Essentially, the key must be random. However, a passphrase generally needs to be easy to remember, so it has significantly less randomness than its length suggests. For example, a 20-letter English phrase, rather than having $20 \times 8 = 150$ bits of randomness, only has about $20 \times 2 = 40$ bits of randomness.

So, most cryptographic software will convert a passphrase into a key through a process called "hashing" or "key initialization." Avoid cryptosystems that skip this phase by using a password directly as a key.

Avoid anything that doesn't let you generate your own keys (e.g., the vendor sends you keys in the mail, or keys are embedded in the copy of the software you buy).

2.5 Implementation Environment

Other factors that can influence the relative security of a product are related to its environment. For example, in software-based encryption packages, is there any plaintext that's written to disk (perhaps in temporary files)? What about operating systems that have the ability to swap processes out of memory on to disk? When something to be encrypted has its plaintext counterpart deleted, is the extent of its deletion a standard removal of its name from the directory contents, or has it been written over? If it's been written over, how well has it been written over? Is that level of security an issue for you? Are you storing cryptographic keys on a multi-user machine? The likelihood of having your keys illicitly accessed is much higher, if so. It's important to consider such things when trying to decide how secure something you implement is (or isn't) going to be.

3 Snake Oil Warning Signs

3.1 "Trust Us, We Know What We're Doing"

Perhaps the biggest warning sign of all is the "trust us, we know what we're doing" message that's either stated directly or implied by the vendor. If the vendor is concerned about the security of their system after describing exactly how it works, it is certainly worthless. Regardless of whether or not they tell, smart people will be able to figure it out. The bad guys after your secrets (especially if you are an especially attractive target, such as a large company, bank, etc.) are not stupid. They will figure out the flaws. If the vendor won't tell you exactly and clearly what's going on inside, you can be sure that they're hiding something, and that the only one to suffer as a result will be you, the customer.

3.2 Technobabble

If the vendor's description appears to be confusing nonsense, it may very well be so, even to an expert in the field. One sign of technobabble is a description which uses newly invented terms or trademarked terms without actually explaining how the system works. Technobabble is a good way to confuse a potential user and to mask the vendor's own lack of expertise.

And consider this: if the marketing material isn't clear, why expect the instruction manual to be any better? Even the best product can be useless if it isn't applied properly. If you can't understand what a vendor is saying, you're probably better off finding something that makes more sense.

3.3 Secret Algorithms

Avoid software which uses secret algorithms. This is not considered a safe means of protecting data. If the vendor isn't confident that its encryption method can withstand scrutiny, then you should be wary of trusting it.

A common excuse for not disclosing an algorithm is that "hackers might try to crack the program's security." While this may be a valid concern, it should be noted that such "hackers" can reverse-engineer the program to see how it works anyway. This is not a problem if the algorithm is strong and the program is implemented properly.

Using a well-known trusted algorithm, providing technical notes explaining the implementation, and making the source code available are signs that a vendor is confident about its product's security. You can take the implementation apart and test it yourself. Even if the algorithm is good, a poor implementation will render a cryptography product completely useless. However, a lock that attackers can't break even when they can see its internal mechanisms is a strong lock indeed. Good cryptography is exactly this kind of lock.

Note that a vendor who specializes in cryptography may have a proprietary algorithm which they will reveal only under a non-disclosure agreement. The crypto product may be perfectly adequate if the vendor is reputable. (But how does a non-expert know if a vendor is reputable in cryptography?) In general, you're best off avoiding secret algorithms.

3.4 Revolutionary Breakthroughs

Beware of any vendor who claims to have invented a "new type of cryptography" or a "revolutionary breakthrough." True breakthroughs are likely to show up in research literature, and professionals in the field typically won't trust them until after years of analysis, when they're not so new anymore.

The strength of any encryption scheme is only proven by the test of time. New crypto is like new pharmaceuticals, not new cars. And in some ways it's worse: if a pharmaceutical company produces bogus drugs, people will start getting sick, but if you're using bogus crypto, you probably won't have any indication that your secrets aren't as secret as you think.

Avoid software which claims to use 'new paradigms' of computing such as cellular automata, neural nets, genetic algorithms, chaos theory, etc. Just because software uses a different method of computation doesn't make it more secure. (In fact, these techniques are the subject of ongoing cryptographic research, and nobody has published successful results based on their use yet.)

Also be careful of specially modified versions of well-known algorithms. This may intentionally or unintentionally weaken the cipher.

It's important to understand the difference between a new cipher and a new product. Engaging in the practice of developing ciphers and cryptographic products is a fine thing to do. However, to do both at the same time is foolish. Many snake oil vendors brag about how they do this, despite the lack of wisdom in such activity.

3.5 Experienced Security Experts, Rave Reviews, and Other Useless Certificates

Beware of any product that claims it was analyzed by "experienced security experts" without providing references. Always look for the bibliography. Any cipher that they're using should appear in a number of scholarly references. If not, it's obviously not been tested well enough to prove or disprove its security.

Don't rely on reviews in newspapers, magazines, or television shows, since they generally don't have cryptographers to analyze software for them. (Celebrity "hackers" who know telephone systems are not necessarily crypto experts.)

Just because a vendor is a well known company or the algorithm is patented doesn't make it secure either.

3.6 Unbreakability

Some vendors will claim their software is "unbreakable." This is marketing hype, and a common sign of snake oil. No algorithm is unbreakable. Even the best algorithms are susceptible to brute-force attacks, though this can be impractical if the key is large enough.

Some companies that claim unbreakability actually have serious reasons for saying so. Unfortunately, these reasons generally depend on some narrow definition of what it means to "break" security. For example, one-time pads (see the next section) are technically unbreakable as far as secrecy goes, but only if several difficult and important conditions are true. Even then, they are trivially vulnerable to known plaintext attacks on the message's integrity. Other systems may be unbreakable only if one of the communicating devices (such as a laptop) isn't stolen. So be sure to find out exactly what the "unbreakable" properties of the system are, and see if the more breakable parts of the system also provide adequate security.

Often, less-experienced vendor representatives will roll their eyes and say, "Of course it's not unbreakable if you do such-and-such." The point is that the exact nature of "such and such" will vary from one product to another. Pick the one that best matches your operational needs without sacraficing your security requirements.

3.7 One-Time-Pads

A vendor might claim the system uses a one-time-pad (OTP), which is provably unbreakable. Technically, the encrypted output of an OTP system is equally likely to decrypt to any same-size plaintext. For example,

```
598v *$_+~ xCtMB0
```

has an equal chance of decrypting to any of these:

```
the answer is yes
the answer is no!
you are a weenie!
```

Snake oil vendors will try to capitalize on the known strength of an OTP. But it is important to understand that any variation in the implementation means that it is not an OTP and has nowhere near the security of an OTP.

An OTP system works by having a "pad" of random bits in the possession of both the sender and recipient, but absolutely no one else. Originally, paper pads were used before general-purpose computers came into being. The pad must be sent from one party to the other securely, such as in a locked briefcase handcuffed to the carrier.

To encrypt an n-bit message, the next n bits in the pad are used as a key. After the bits are used from the pad, they're destroyed, and can *never* be used again.

The bits in the pad cannot be generated by an algorithm or cipher. They must be truly random, using a real random source such as specialized hardware, radioactive decay timings, etc. Some snake oil vendors will try to dance around this issue, and talk about functions they perform on the bit stream, things they do with the bit stream vs. the plaintext, or something similar. But this still doesn't change the fact that anything that doesn't use *real* random bits is not an OTP. The important part of an OTP is the source of the bits, not what one does with them.

OTPs are seriously vulnerable if you ever reuse a pad. For instance, the NSA's VENONA project [4], without the benefit of computer assistance, managed to decrypt a series of KGB messages encrypted with faulty pads. It doesn't take much work to crack a reused pad.

The real limitation to practical use of OTPs is the generation and distribution of truly random keys. You have to distribute at least one bit of key for every bit of data transmitted. So OTPs are awkward for general purpose cryptography. They're only practical for extremely-low-bandwidth communication channels where two parties can exchange pads with a method different than they exchange messages. (It is rumored that a link from Washington, D.C., to Moscow was encrypted with an OTP.)

Further, if pads are provided by a vendor, you cannot verify the quality of the pads. How do you know the vendor isn't sending the same bits to everyone? Keeping a copy for themselves? Or selling a copy to your rivals?

Also, some vendors may try to confuse random session keys or initialization vectors with OTPs.

3.8 Algorithm or product X is insecure

Be wary of anything that claims that competing algorithms or products are insecure without providing evidence for these claims. Sometimes attacks are theoretical or impractical, requiring special circumstances or massive computing power over many years, and it's easy to confuse a layman by mentioning these.

3.9 Recoverable Keys

If there is a key-backup or key-escrow system, are you in control of the backup or does someone else hold a copy of the key? Can a third party recover your key without much trouble? Remember, you have no security against someone who has your key.

If the vendor claims it can recover lost keys without using some type of key-escrow service, avoid it. The security is obviously flawed.

3.10 Exportable from the USA

If the software is made in the USA, can it be exported? Strong cryptography is considered dangerous munitions by the United States and requires approval from the US Bureau of Export Administration, under the US Department of Commerce, before it can leave the country. Various interested government agencies serve as consultants to the Bureau of Export Administration when evaluating such requests. (The U.S. isn't alone in this; some other nations have similar export restrictions on strong cryptography.) Chances are, if the software has been approved for export, the algorithm is weak or crackable.

If the vendor is unaware of export restrictions, avoid their software. For example, if they claim that the IDEA cipher can be exported when most vendors (and the US Government!) do not make such a claim, then the vendor is probably lacking sufficient clue to provide you with good cryptography.

Because of export restrictions, some decent crypto products come in two flavors: US-only and exportable. The exportable version will be crippled, probably by using smaller keys, making it easy to crack.

There are no restrictions on *importing* crypto products into the US, so a non-US vendor can legally offer a single, secure version of a product for the entire world.

Note that a cryptosystem may not be exportable from the US even if it is available outside the US: sometimes a utility is illegally exported and posted on an overseas site.

3.11 "Military Grade"

Many crypto vendors claim their system is "military grade." This is a meaningless term, since there isn't a standard that defines "military grade," other than actually being used by various armed forces. Since these organizations don't reveal what crypto they use, it isn't possible to prove or disprove that something is "military grade."

Unfortunately, some good crypto products also use this term. Watch for this in combination with other snake oil indicators, e.g., "our military-grade encryption system is exportable from the US!"

4 Other Considerations

Avoid vendors who don't seem to understand anything described in the "Basic Concepts" section above.

Avoid anything that allows someone with your copy of the software to access files, data, etc. without needing some sort of key or passphrase.

Beware of products that are designed for a specific task, such as data archiving, and have encryption as an additional feature. Typically, it's better to use an encryption utility for encryption, rather than some tool designed for another purpose that adds encryption as an afterthought.

No product is secure if used improperly. You can be the weakest link in the chain if you use a product carelessly. Do not trust any product to be foolproof, and be wary of any product that claims it is.

Interface isn't everything: user-friendliness is important, but be wary of anything that puts too much emphasis on ease of use without due consideration to cryptographic strength.

5 Glossary

algorithm A procedure or mathematical formula. Cryptographic algorithms convert plaintext to and from ciphertext.

cipher Synonym for "cryptographic algorithm"

cryptanalysis To solve or "break" a cryptosystem.

EAR Export Administration Regulations. The rules under which the export of cryptographic software from the US are governed now.

escrow A third party able to decrypt messages sent from one person to another. Although this term is often used in connection with the US Government's "Clipper" proposals, it isn't limited to government-mandated ability to access encrypted information at will. Some corporations might wish to have their employees use cryptosystems with escrow features when conducting the company's business, so the information can be retrieved should the employee be unable to unlock it himself later, (if he were to forget his passphrase, suddenly quit, get run over by a bus, etc.) Or, someone might wish his spouse or lawyer to be able to recover encrypted data, etc., in which case he could use a cryptosystem with an escrow feature.

initialization vector One of the problems with encrypting such things as files in specific formats (i.e., that of a word processor, email, etc.) is that there is a high degree of predictability about the first bytes of the message. This could be used to break the encrypted message easier than by brute force. In ciphers where one block of data is used to influence the ciphertext of the next (such as CBC), a random block of data is encrypted and used as the first block of the encrypted message, resulting in a less predictable ciphertext message. This random block is known as the initialization vector. The decryption process also performs the function of removing the first block, resulting in the original plaintext.

ITAR International Traffic in Arms Regulations. These are the rules by which munitions, as defined by the US State Department, may (or may not) be exported from the US. Until recently, this also included the export of cryptography. The exportability of cryptography is now in the hands of the Bureau of Export Administration, under the US Department of Commerce.

key A piece of data that, when fed to an algorithm along with ciphertext, will yield plaintext. (Or, when fed to an algorithm along with plaintext, will yield ciphertext.

random session key This is a temporary key that is generated specifically for one message. Typically, in public key cryptosystems, the message to be sent is encrypted with a symmetric key that was specifically generated for that message. The encrypted version of that message, as well as the associated session key can then be encrypted with the recipient's public key. When the recipient decrypts the message, then, the system will actually decrypt the message it gets (which is the ciphertext message and the symmetric key to decrypt it), and then use the symmetric key to decrypt the ciphertext. The result is the plaintext message. This is often done because of the tremendous difference in the speed of symmetric vs. asymmetric ciphers.

References

[1] B. Schneier. *Applied Cryptography*, 2e. John Wiley & Sons. 1996.

[2] M. Blaze, W. Diffie, R. L. Rivest, B. Schneier, T. Shimomura, E. Thompson, M. Wiener. "Minimal Key Lengths for Symmetric Ciphers to Provide Adequate Commercial Security". Available at ftp://ftp.research.att.com/dist/mab/keylength.ps and http://theory.lcs.mit.edu/˜ rivest/bsa-final-report.ascii

[3] The Crypt Cabal. *Cryptography FAQ*. Available at http://www.cis.ohio-state.edu/hypertext/faq/usenet/cryptography-faq/top.html.

[4] The National Security Agency. *The VENONA Project*. Available at http://www.nsa.gov/docs/venona/venona.html.

[5] RSA Data Security, Inc. *1997 Secret Key Challenge*. Available at http://www.rsa.com/rsalabs/97challenge/.