Network Working Group
Request for Comments: 4772
Category: Informational
S. Kelly
Aruba Networks
December 2006

Security Implications of Using the Data Encryption Standard (DES)

Status of This Memo

This memo provides information for the Internet community. It does not specify an Internet standard of any kind. Distribution of this memo is unlimited.

Copyright Notice

Abstract

The Data Encryption Standard (DES) is susceptible to brute-force attacks, which are well within the reach of a modestly financed adversary. As a result, DES has been deprecated, and replaced by the Advanced Encryption Standard (AES). Nonetheless, many applications continue to rely on DES for security, and designers and implementers continue to support it in new applications. While this is not always inappropriate, it frequently is. This note discusses DES security implications in detail, so that designers and implementers have all the information they need to make judicious decisions regarding its use.

Table of Contents

1. Introduction

The Data Encryption Standard [DES] is the first encryption algorithm approved by the U.S. government for public disclosure. Brute-force attacks became a subject of speculation immediately following the algorithm's release into the public sphere, and a number of researchers published discussions of attack feasibility and explicit brute-force attack methodologies, beginning with [DH77].

In the early to mid 1990s, numerous additional papers appeared, including Wiener's "Efficient DES Key Search" [WIEN94], and "Minimal Key Lengths for Symmetric Ciphers to Provide Adequate Commercial Security" [BLAZ96]. While these and various other papers discussed the theoretical aspects of DES-cracking machinery, none described a specific implementation of such a machine. In 1998, the Electronic Frontier Foundation (EFF) went much further, actually building a device and freely publishing the implementation details for public review [EFF98].

Despite the fact that the EFF clearly demonstrated that DES could be brute-forced in an average of about 4.5 days with an investment of less than $250,000 in 1998, many continue to rely on this algorithm even now, more than 8 years later. Today, the landscape is significantly different: DES can be broken by a broad range of attackers using technologies that were not available in 1998, including cheap Field Programmable Gate Arrays (FPGAs) and botnets [BOT05]. These and other attack methodologies are described in detail below.

Given that the Advanced Encryption Standard [AES] has been approved by the U.S. government (under certain usage scenarios) for top-secret applications [AES-NSA], and that triple DES (3DES) is not susceptible to these same attacks, one might wonder: why even bother with DES anymore? Under more ideal circumstances, we might simply dispense with it, but unfortunately, this would not be so simple today. DES has been widely deployed since its release in the 1970s, and many systems rely on it today. Wholesale replacement of such systems would be very costly. A more realistic approach entails gradual replacement of these systems, and this implies a term of backward compatibility support of indefinite duration.

In addition to backward compatibility, in isolated instances there may be other valid arguments for continued DES support. Still, reliance upon this deprecated algorithm is a serious error from a security design perspective in many cases. This note aims to clarify the security implications of this choice given the state of technology today, so that developers can make an informed decision as to whether or not to implement this algorithm.

1.1. Executive Summary of Findings and Recommendations

For many years now, DES usage has been actively discouraged by the security area of the IETF, but we nevertheless continue to see it in use. Given that there are widely published accounts of real attacks and that we have been vocally discouraging its use, a question arises: why aren't people listening? We can only speculate, but one possibility is that they simply do not understand the extent to which DES has been marginalized by advancing cryptographic science and technology. Another possibility is that we have not yet been appropriately explicit and aggressive about this. With these particular possibilities in mind, this note sets out to dispel any remaining illusions.

The depth of background knowledge required to truly understand and fully appreciate the security risks of using DES today is somewhat daunting, and an extensive survey of the literature suggests that there are very few published materials encompassing more than a fraction of the considerations all in one place, with [CURT05] being one notable exception. However, even that work does not gather all of the pieces in such a way as to inform an implementer of the current real-world risks, so here we try to fill in any remaining gaps.

For convenience, the next section contains a brief summary of recommendations. If you don't know the IETF's current position on DES, and all you want is a summary, you may be content to simply read the recommendation summary section, and skip the rest of the document. If you want a more detailed look at the history and current state-of-the-art with respect to attacking DES, you will find that in subsequent sections.

1.1.1. Recommendation Summary

There are several ways to attack a cryptographic algorithm, from simple brute force (trying each key until you find the right one) to more subtle cryptanalytic approaches, which take into account the internal structure of the cipher. As noted in the introduction, a dedicated system capable of brute-forcing DES keys in less than 5 days was created in 1998. Current "Moore's Law" estimates suggest that a similar machine could be built today for around $15,000 or less, and for the cost of the original system (~$250,000) we could probably build a machine capable of cracking DES keys in a few hours.

Additionally, there have been a number of successful distributed attacks on DES [CURT05], and with the recent arrival of botnets [BOT05], these results are all the more onerous. Furthermore, there are a number of cryptanalytic attacks against DES, and while some of these remain purely theoretical in nature at present, at least one was recently implemented using a FPGA that can deduce a DES key in 12-15 hours [FPL02]. Clearly, DES cannot be considered a "strong" cryptographic algorithm by today's standards.

To summarize current recommendations on using DES, the simple answer is "don't use it - it's not safe." While there may be use cases for which the security of DES would be sufficient, it typically requires a security expert to determine when this is true. Also, there are much more secure algorithms available today (e.g., 3DES, AES) that are much safer choices. The only general case in which DES should still be supported is when it is strictly required for backward compatibility, and when the cost of upgrading outweighs the risk of exposure. However, even in these cases, recommendations should probably be made to phase out such systems.

If you are simply interested in the current recommendations, there you have it: don't use DES. If you are interested in understanding how we arrive at this conclusion, read on.

2. Why Use Encryption?

In order to assess the security implications of using DES, it is useful and informative to review the basic rationale for using encryption. In general, we encrypt information because we desire confidentiality. That is, we want to limit access to information, to keep something private or secret. In some cases, we want to share the information within a limited group, and in other cases, we may want to be the sole owner of the information in question.

Sometimes, the information we want to protect has value only to the individual (e.g., a diary), and a loss of confidentiality, while potentially damaging in some limited ways, would typically not be catastrophic. In other cases, the information might have significant financial implications (e.g., a company's strategic marketing plan). And in yet others, lives could be at stake.

In order to gauge our confidentiality requirements in terms of encryption strength, we must assess the value of the information we are trying to protect, both to us and to a potential attacker. There are various metrics we can employ for this purpose:

- o Cost of confidentiality loss: What could we lose if an adversary were to discover our secret? This gives some measure of how much effort we should be willing to expend to protect the secret.

- o Value to adversary: What does the attacker have to gain by discovering our secret? This gives some measure of how much an adversary might reasonably be willing to spend to learn the secret.

- o Window of opportunity: How long does the information have value to an adversary? This gives some measure of how acceptable a weakness might be. For example, if the information is valuable to an attacker for months and it takes only days to break the encryption, we probably need much stronger encryption. On the other hand, if the window of opportunity is measured in seconds, then an encryption algorithm that takes days to break may be acceptable.

There are certainly other factors we would consider in conducting a comprehensive security analysis, but these are enough to give a general sense of important questions to answer when evaluating DES as a candidate encryption algorithm.

3. Real-World Applications and Threats

Numerous commonly used applications rely on encryption for confidentiality in today’s Internet. To evaluate the sufficiency of a given cryptographic algorithm in this context, we should begin by asking some basic questions: what are the real-world risks to these applications, i.e., how likely is it that an application might actually be attacked, and by whom, and for what reasons?

While it is difficult to come up with one-size-fits-all answers based on general application descriptions, we can easily get some sense of the relative threat to many of these applications. It is important to note that what follows is not an exhaustive enumeration of all likely threats and attacks, but rather, a sampling that illustrates that real threats are more prevalent than intuition might suggest.

Here are some examples of common applications and related threats:

- o Site-to-site VPNs: Often, these are used to connect geographically separate corporate offices. Data traversing such links is often business critical, and sometimes highly confidential. The FBI estimates that every year, billions of U.S. dollars are lost to foreign competitors who deliberately target economic intelligence in U.S. industry and technologies [FBI06]. Searching for ’corporate espionage’ in Google yields many interesting links, some of which indicate that foreign competitors are not the only threat to U.S. businesses. Obviously, this threat can be generalized to include businesses of any nationality.

- o Remote network access for business: See previous item.

- o Webmail/email encryption: See Site-to-site VPNs.

- o Online banking: Currently, the most common threat to online banking is in the form of "phishing", which does not rely on breaking session encryption, but instead relies on tricking users into providing their account information. In general, direct attacks on session encryption for this application do not scale well. However, if a particular bank were known to use a weak encryption algorithm for session security, it might become worthwhile to develop a broader attack against that bank. Given that organized criminal elements have been found behind many phishing attacks, it is not difficult to imagine such scenarios.

- o Electronic funds transfers (EFTs): The ability to replay or otherwise modify legitimate EFTs has obvious financial incentives (and implications). Also, an industrial spy might see a great deal of intelligence value in the financial transactions of a target company.

- o Online purchases (E-commerce): The FBI has investigated a number of organized attacks on e-commerce applications [FBI01]. If an attacker has the ability to monitor e-commerce traffic directed to a large merchant that relies on weak encryption, the attacker could harvest a great deal of consumer credit information. This is the sort of data "phishers" currently harvest on a much smaller scale, so one can easily imagine the value of such a target.

- o Internet-based VoIP applications (e.g., Skype): While many uses of this technology are innocuous (e.g., long distance calls to family members), VoIP technology is also used for business purposes (see discussion of FBI estimates regarding corporate espionage above).

- o Cellular telephony: Cell phones are very common, and are frequently used for confidential conversations in business, medicine, law enforcement, and other applications.

- o Wireless LAN: Wireless technology is used by many businesses, including the New York Stock Exchange [NYSE1]. The financial incentives for an attacker are significant in some cases.

- o Personal communications (e.g., secure instant messaging): Such communication may be used for corporate communications (see industrial espionage discussion above), and may also be used for financial applications such as stock/securities trading. This has both corporate/industrial espionage and financial implications.

- o Laptop hard-drive encryption: See discussion on corporate/ industrial espionage above. Also, consider that stolen and lost laptops have been cited for some of the more significant losses of control over sensitive personal information in recent years, notably the Veterans Affairs data loss [VA1].

There are real-world threats to everyday encryption applications, some of which could be very lucrative to an attacker (and by extension, very costly to the victim). It is important to note that if some of these attacks are infrequent today, it is precisely because the threats are recognized, and appropriately strong cryptographic algorithms are used. If "weak" cryptographic algorithms were to be used instead, the implications are indeed thought-provoking.

In keeping with the objectives of this document, it is important to note that the U.S. government has never approved the use of DES for anything but unclassified applications. While DES is still approved for unclassified uses until May 19, 2007, the U.S. government clearly sees the need to move to higher ground. For details on the National Institute of Standards and Technology (NIST) DES Transition plan, see [NIST-TP]. Despite this fact, DES is still sometimes chosen to protect some of the applications described above. Below, we discuss why this should, in many cases, be remedied.

4. Attacking DES

DES is a 64-bit block cipher having a key size of 56 bits. The key actually has 64 bits (matching the block size), but 1 bit in each byte has been designated a ’parity’ bit, and is not used for cryptographic purposes. For a full discussion of the history of DES along with an accessible description of the algorithm, see [SCHN96].

A detailed description of the various types of attacks on cryptographic algorithms is beyond the scope of this document, but for clarity, we provide the following brief descriptions. There are two general aspects of attacks we must consider: the form of the inputs/outputs along with how we might influence them, and the internal function of the cryptographic operations themselves.

In terms of input/output form, some of the more commonly discussed attack characteristics include the following:

- o known plaintext - the attacker knows some of the plaintext corresponding to some of the ciphertext

- o ciphertext-only - only ciphertext is available to the attacker, who has little or no information about the plaintext

- o chosen plaintext - the attacker can choose which plaintext is encrypted, and obtain the corresponding ciphertext

- o birthday attacks - relies on the fact that for N elements, collisions can be expected in ~sqrt(N) randomly chosen samples; for systems using CBC mode with random Initialization Vectors (IVs), ciphertext collisions can be expected in about 2^28 samples. Such collisions leak information about the corresponding plaintexts: if the same cryptographic key is used, then the xor of the IVs is equal to the xor of the plaintexts.

- o meet-in-the-middle attacks - leverages birthday characteristic to precompute potential key collision values

Due to the limited scope of this document, these are very brief descriptions of very complex subject matter. For more detailed discussions on these and many related topics, see [SCHN96], [HAC], or [FERG03].

As for attack characteristics relating to the operational aspects of cipher algorithms, there are essentially two broad classes we consider: cryptanalytic attacks, which exploit some internal structure or function of the cipher algorithm, and brute-force attacks, in which the attacker systematically tries keys until the right one is found. These could alternatively be referred to as white box and black box attacks, respectively. These are discussed further below.

4.1. Brute-Force Attacks

In general, a brute-force attack consists of trying each possible key until the correct key is found. In the worst case, this will require 2^n steps for a key size of n bits, and on average, it will require 2^n-1 steps. For DES, this implies 2^56 encryption operations in the worst case, and 2^55 encryption operations on average, if we assume no shortcuts exist. As it turns out, the complementation property of DES provides an attack that yields a reduction by a factor of 2 for a chosen plaintext attack, so this attack requires an average of 2^54 encryption operations.

Above, we refer to 2^n 'steps'; note that what a 'step' entails depends to some extent on the first attack aspect described above, i.e., what influence and knowledge we have with respect to input/ output forms. Remember, in the worst case, we will be performing 72,057,594,037,927,936 -- over 72 quadrillion -- of these 'steps'. In the most difficult case, we have ciphertext only, and no knowledge of the input, and this is very important.

If the input is effectively random, we cannot tell by simply looking at a decrypted block whether we've succeeded or not. We may have to resort to other potentially expensive computation to make this determination. While the effect of any additional computation will be linear across all keys, repeating a large amount of added computation up to 72 quadrillion times could have a significant impact on the cost of a brute-force attack against the algorithm. For example, if it takes 1 additional microsecond per computation, this will add almost 101 days to our worst-case search time, assuming a serial key search.

On the other hand, if we can control the input to the encryption function (known plaintext), we know precisely what to expect from the decryption function, so detecting that we've found the key is straightforward. Alternatively, even if we don't know the exact input, if we know something about it (e.g., that it's ASCII), with limited additional computation we can infer that we've most likely found a key. Obviously, which of these conditions holds may significantly influence attack time.

4.1.1. Parallel and Distributed Attacks

Given that a brute-force attack involves systematically trying keys until we find the right one, it is obviously a good candidate for parallelization. If we have N processors, we can find the key roughly N times faster than if we have only 1 processor. This requires some sort of centralized control entity that distributes the work and monitors the search process, but is quite straightforward to implement.

There are at least two approaches to parallelization of a brute-force attack on a block cipher: the first is to build specialized high-speed hardware that can rapidly cycle through keys while performing the cryptographic and comparison operations, and then replicate that hardware many times, while providing for centralized control. The second involves using many copies of general purpose hardware (e.g., a PC), and distributing the load across these while placing them under the control of one or more central systems. Both of these approaches are discussed further in sections 5 and 6.

4.2. Cryptanalytic Attacks

Brute-force attacks are so named because they don't require much intelligence in the attack process -- they simply try one key after the other, with little or no intelligent keyspace pruning. Cryptanalytic attacks, on the other hand, rely on application of some intelligence ahead of time, and by doing so, provide for a significant reduction of the search space.

While an in-depth discussion of cryptanalytic techniques and the resulting attacks is well beyond the scope of this document, it is important to briefly touch on this area in order to set the stage for subsequent discussion. It is also important to note that, in general, cryptanalysis can be applied to any cryptographic algorithm with varying degrees of success. However, we confine ourselves here to discussing specific results with respect to DES.

Here is a very brief summary of the currently known cryptanalytic attacks on DES:

- o Differential Cryptanalysis - First discussed by Biham and Shamir, this technique (putting it very simply) analyzes how differences in plaintext correspond to differences in ciphertext. For more detail, see [BIH93].

- o Linear Cryptanalysis - First described by Matsui, this technique uses linear approximations to describe the internal functions of DES. For more detail, see [MAT93].

- o Interpolation Attack - This technique represents the S-boxes of DES with algebraic functions, and then estimates the coefficients of the functions. For more information, see [JAK97].

- o Key Collision Attack - This technique exploits the birthday paradox to produce key collisions [BIH96].

- o Differential Fault Analysis - This attack exploits the electrical characteristics of the encryption device, selectively inducing faults and comparing the results with uninfluenced outputs. For more information, see [BIH96-2].

Currently, the best publicly known cryptanalytic attacks on DES are linear and differential cryptanalysis. These attacks are not generally considered practical, as they require 2^43 and 2^47 known plaintext/ciphertext pairs, respectively. To get a feel for what this means in practical terms, consider the following:

- o For linear cryptanalysis (the more efficient of the two attacks), the attacker must pre-compute and store 2^43 ciphertexts; this requires 8,796,093,022,208 (almost 9 trillion) encryption operations.

- o Each ciphertext block is 8 bytes, so the total required storage is 70,368,744,177,664 bytes, or about 70,369 gigabytes of storage. If the plaintext blocks cannot be automatically derived, they too must be stored, potentially doubling the storage requirements.

- o The 2^43 known plaintext blocks must be somehow fed to the device under attack, and that device must not change the encryption key during this time.

Clearly, there are practical issues with this attack. Still, it is sobering to look at how much more realistic 70,000 gigabytes of storage is today than it must have seemed in 1993, when Matsui first proposed this attack. Today, 400-GB hard drives can be had for around $0.35/gigabyte. If we only needed to store the known ciphertext, this amounts to ~176 hard drives at a cost of less than $25,000. This is probably practical with today's technology for an adversary with significant financial resources, though it was difficult to imagine in 1993. Still, numerous other practical issues remain.

4.3. Practical Considerations

Above, we described several types of attacks on DES, some of which are more practical than others, but it's very important to recognize that brute force represents the very worst case, and cryptanalytic attacks can only improve on this. If a brute-force attack against a given DES application really is feasible, then worrying about the practicality of the other theoretical attack modes is just a distraction. The bottom line is this: if DES can be brute-forced at a cost the attacker can stomach today, this cost will invariably come down as technology advances.

5. The EFF DES Cracker

On the question as to whether DES is susceptible to brute-force attack from a practical perspective, the answer is a resounding and unequivocal "yes". In 1998, the Electronic Frontier Foundation financed the construction of a "DES Cracker", and subsequently published "Cracking DES" [EFF98]. For a cost of less than $250,000, this system can find a 56-bit DES key in the worst-case time of around 9 days, and in 4.5 days on average.

Quoting from [EFF98],

"The design of the EFF DES Cracker is simple in concept. It consists of an ordinary personal computer connected with a large array of custom chips. Software in the personal computer instructs the custom chips to begin searching, and interacts with the user. The chips run without further help from the software until they find a potentially interesting key, or need to be directed to search a new part of the key space. The software periodically polls the chips to find any potentially interesting keys that they have turned up.

The hardware's job isn't to find the answer. but rather to eliminate most of the answers that are incorrect. Software is then fast enough to search the remaining potentially-correct keys, winnowing the false positives from the real answer. The strength of the machine is that it replicates a simple but useful search circuit thousands of times, allowing the software to find the answer by searching only a tiny fraction of the key space.

As long as there is a small bit of software to coordinate the effort, the problem of searching for a DES key is 'highly parallelizable'. This means the problem can be usefully solved by many machines working in parallel, simultaneously. For example, a single DES-Cracker chip could find a key by searching for many years. A thousand DES-Cracker chips can solve the same problem in one thousandth of the time. A million DES-Cracker chips could theoretically solve the same problem in about a millionth of the time, though the overhead of starting each chip would become visible in the time required. The actual machine we built contains 1536 chips."

This project clearly demonstrated that a practical system for brute force DES attacks was well within reach of many more than previously assumed. Practically any government in the world could easily produce such a machine, and in fact, so could many businesses. And that was in 1998; the technological advances since then have greatly reduced the cost of such a device. This is discussed further below.

6. Other DES-Cracking Projects

In the mid-1990s, many were interested in whether or not DES was breakable in a practical sense. RSA sponsored a series of DES Challenges over a 3-year period beginning January of 1997. These challenges were created in order to help underscore the point that cryptographic strength limitations imposed by the U.S. government's export policies were far too modest to meet the security requirements of many users.

The first DES challenge was solved by the DESCHALL group, led by Rocke Verser, Matt Curtin, and Justin Dolske [CURT05][RSA1]. They created a loosely-knit distributed effort staffed by volunteers and backed by Universities and corporations all over the world who donated their unused CPU cycles to the effort. They found the key in 90 days.

The second DES challenge was announced on December 19, 1997 [RSA2][CURT05], and on February 26, 1998, RSA announced a winner. This time, the challenge was solved by group called distributed.net working together with the EFF, in a total of 39 days [RSA3] [CURT05]. This group coordinated 22,000 participants and over 50,000 CPUs.

The third DES challenge was announced on December 22, 1998 [RSA4][CURT05], and on January 19, 1999, RSA announced the winner. This time, the challenge was again solved by distributed.net working together with the EFF, in a total of 22 hours [RSA5]. This was a dramatic improvement over the second challenge, and should give some idea of where we're headed with respect to DES.

7. Building a DES Cracker Today

We've seen what was done in the late 1990s -- what about today? A survey of the literature might lead one to conclude that this topic is no longer interesting to cryptographers. Hence, we are left to infer the possibilities based on currently available technologies. One way to derive an approximation is to apply a variation on "Moore's Law": assume that the cost of a device comparable to the one built by the EFF would be halved roughly every N months. If we take N=18, then for a device costing $250,000 at the end of 1998, this would predict the following cost curve:

- o mid-2000............: $125,000
- o beginning of 2002...: $62,500
- o mid-2003............: $31,250
- o beginning of 2006...: $15,625

It's important to note that strictly speaking, "Moore's Law" is more an informal approximation than a law, although it has proven to be uncannily accurate over the last 40 years or so. Also, some would disagree with the use of an 18-month interval, preferring a more conservative 24 months instead. So, these figures should be taken with the proverbial grain of salt. Still, it's important to recognize that this is the cost needed not to crack one key, but to get into the key-cracking business. Offering key-cracking services and keeping the machine relatively busy would dramatically decrease the cost to a few hundred dollars per unit or less.

Given that such calculations roughly hold for other computing technologies over the same time interval, the estimate above does not seem too unreasonable, and is probably within a factor of two of today's costs. Clearly, this would seem to indicate that DES-cracking hardware is within reach of a much broader group than in 1998, and it is important to note that this assumes no design or algorithm improvements since then.

To put this in a slightly different light, let's consider the typical rendition of Moore's Law for such discussions. Rather than considering shrinking cost for the same capability, consider instead increasing capability for the same cost (i.e., doubling circuit densities every N months). Again choosing N=18, our DES-cracking capability (in worst-case time per key) could be expected to have approximately followed this performance curve over the last 7 or so years:

- 1998................: 9 days
- mid-2000............: 4.5 days
- beginning of 2002...: 2.25 days
- mid-2003............: 1.125 days
- beginning of 2006...: 0.5625 days

That's just over a half-day in the worst case for 2006, and under 7 hours on average. And this, for an investment of less than $250,000. It's also very important to note that we are talking about worst-case and average times here - sometimes, keys will be found much more quickly. For example, using such a machine, 1/4 of all possible DES keys will be found within 3.375 hours. 1/8 of the keys will be found in less than 1 hour and 42 minutes. And this assumes no algorithmic improvements have occurred. And again, this is an estimate; your actual mileage may vary, but the estimate is probably not far from reality.

7.1. FPGAs

Since the EFF device first appeared, Field Programmable Gate Arrays (FPGAs) have become quite common, and far less costly than they were in 1998. These devices allow low-level logic programming, and are frequently used to prototype new logic designs prior to the creation of more expensive custom chips (also known as Application Specific Integrated Circuits, or ASICs). They are also frequently used in place of ASICs due to their lower cost and/or flexibility. In fact, a number of embedded systems implementing cryptography have employed FPGAs for this purpose.

Due to their generalized nature, FPGAs are naturally slower than ASICs. While the speed difference varies based on many factors, it is reasonable for purposes of this discussion to say that well-designed FPGA implementations typically perform cryptographic operations at perhaps 1/4 the speed of well-designed ASICs performing the same operations, and sometimes much slower than that. The significance of this comparison will become obvious shortly.

In our Moore's Law estimate above, we noted that the cost extrapolation assumes no design or algorithm improvements since 1998. It also implies that we are still talking about a brute-force attack. In section 4 ("Attacking DES"), we discussed several cryptanalytic attacks, including an attack that employs linear cryptanalysis [MAT93]. In general, this attack has been considered impractical, but in 2002, a group at Universite Catholique de Louvain in Belgium built a DES cracker based on linear cryptanalysis, which, employing a single FPGA, returns a DES key in 12-15 hours [FPL02].

While there are still some issues of practicality in terms of applying this attack in the real world (i.e., the required number of known plaintext-ciphertext pairs), this gives a glimpse of where technology is taking us with respect to DES attack capabilities.

7.2. ASICs

Application Specific Integrated Circuits are specialized chips, typically optimized for a particular set of operations (e.g., encryption). There are a number of companies that are in the business of designing and selling cryptographic ASICs, and such chips can be had for as little as $15 each at the low end. But while these chips are potentially much faster than FPGAs, they usually do not represent a proportionally higher threat when it comes to DES-cracking system construction.

The primary reason for this is cost: it currently costs more than $1,000,000 to produce an ASIC. There is no broad commercial market for crypto-cracking ASICs, so the number a manufacturer could expect to sell is probably small. Likewise, a single attacker is not likely to require more than a few of these. The bottom line: per-chip costs would be very high; when compared to the costs of FPGAs capable of similar performance, the FPGAs are clear winners. This doesn't mean such ASICs have never been built, but the return is probably not worth the investment for the average attacker today, given the other available options.

7.3. Distributed PCs

Parallel processing is a powerful tool for conducting brute-force attacks against a block cipher. Since each key can be tested independently, the keyspace can easily be carved up and distributed across an arbitrary number of processors, all of which are running identical code. A central "control" processor is required for distributing tasks and evaluating results, but this is straightforward to implement, and this paradigm has been applied to many computing problems.

While the EFF demonstrated that a purpose-built system is far superior to general purpose PCs when applied to cracking DES, the DESCHALL effort [CURT05][RSA1] aptly demonstrated that the idle cycles of everyday users' PCs could be efficiently applied to this problem. As noted above, distributed.net teamed with the EFF group to solve the third RSA DES Challenge using a combination of PCs and the EFF's "Deep Crack" machine to find a DES key in 22 hours. And that was using 1999 technologies.

Clearly, PCs have improved dramatically since 1999. At that time, state-of-the-art desktops ran at around 800MHz. Today, desktop PCs commonly run at 3-4 times that speed, and supporting technologies (memory, cache, storage) offer far higher performance as well. Since the distributed.net effort used a broad spectrum of computers (from early 1990s desktops to state-of-the-art (in 1999) multiprocessors, according to [DIST99]), it is difficult to do a direct comparison with today's technologies. Still, we know that performance has, in general, followed the prediction of Moore's Law, so we should expect an improvement on the order of a factor of 8-16 by now, even with no algorithmic improvements

7.3.1. Willing Participants

It is important to note that the distributed.net efforts have relied upon willing participants. That is, participants must explicitly and voluntarily join the effort. It is equally important to note that only the idle cycles of the enrolled systems are used. Depending on the way in which "idle" is defined, along with the user's habits and computing requirements, this could have a significant effect on the contribution level of a given system.

These factors impose significant limitations in terms of scale. While distributed.net was able to enlist over 100,000 computers from around the world for the third RSA DES Challenge, this is actually a rather small number when compared to 2^56 (over 72 quadrillion) possible DES keys. And when you consider the goal (i.e., to prove DES can be cracked), it seems reasonable to assume these same participants would not willingly offer up their compute cycles for a more nefarious use (like attacking the keys used to encrypt your online banking session). Hence, this particular model does not appear to pose a significant threat to most uses of encryption today. However, below, we discuss a variation on this approach that does pose an immediate threat.

7.3.2. Spyware and Viruses and Botnets (oh my!)

"Spyware" is a popular topic in security newsfeeds these days. Most of these applications are intended to display context-sensitive advertisements to users, and some actually modify a user's web browsing experience, directing them to sites of the distributor's choice in an effort to generate revenue. There are many names for this type of software, but for our purposes, we will refer to it simply as "spyware". And while there are some instances in which rogue software actually does spy on hapless users and report things back to the issuer, we do not focus here on such distinctions.

Indeed, what we are more interested in is the broader modality in which this software functions: it is typically installed without the explicit knowledge and/or understanding of the user, and typically runs without the user's knowledge, sometimes slowing the user's PC to a crawl. One might note that such behavior seems quite surprising in view of the fact that displaying ads to users is actually a light-weight task, and wonder what this software is actually doing with all those compute cycles.

Worms and viruses are also very interesting: like spyware, these are installed without the user's knowledge or consent, and they use the computer in ways the user would not voluntarily allow. And unlike the spyware that is most common today, this malware usually contains explicit propagation technology by which it automatically spreads. It is not difficult to imagine where we are going with this: if you combine these techniques, forcible induction of user machines into an "army" of systems becomes possible. This approach was alluded to in [CURT98] and, in fact, is being done today.

Botnets [BOT05] represent a relatively recent phenomena. Using various propagation techniques, malware is distributed across a range of systems, where it lies in wait for a trigger of some sort. These "triggers" may be implemented through periodic polling of a centralized authority, the arrival of a particular date, or any of a large number of other events. Upon triggering, the malware executes its task, which may involve participating in a Distributed Denial of Service (DDoS) attack, or some other type of activity.

Criminal groups are currently renting out botnets for various uses [CERT01]. While reported occurrences have typically involved using these rogue networks for DDoS attacks, we would be naive to think other uses (e.g., breaking encryption keys) have not been considered. Botnets greatly mitigate the scaling problem faced by distributed.net: it is no longer a volunteer-only effort, and user activity no longer significantly impedes the application's progress. This should give us pause.

It is very important to clearly recognize the implications of this: botnets are cheap, and there are lots of PCs out there. You don't need the $15,625 that we speculated would be enough to build a copy of the EFF system today -- you only need a commodity PC on which to develop the malware, and the requisite skills. Or, you need access to someone with those things, and a relatively modest sum of cash. The game has changed dramatically.

8. Why is DES Still Used?

Obviously, DES is not secure by most measures -- why is it still used today? There are probably many reasons, but here are perhaps the most common:

- o Backward compatibility - Numerous deployed systems support DES, and rather than replace those systems, new systems are implemented with compatibility in mind.

- o Performance - Many early VPN clients provided DES as the default cryptographic algorithm, because PCs of the day suffered a noticeable performance hit when applying stronger cryptography (e.g., 3DES).

- o Ignorance - People simply do not understand that DES is no longer secure for most uses.

While there are probably other reasons, these are the most frequently cited.

Performance arguments are easily dispensed with today. PCs have more than ample power to implement stronger cryptography with no noticeable performance impact, and for systems that are resource constrained, there are strong algorithms that are far better performers than DES (e.g., AES-128). And while backward compatibility is sometimes a valid argument, this must be weighed carefully. At the point where the risk is higher than the cost of replacement, legacy systems should be abandoned.

With respect to the third reason (ignorance), this note attempts to address this, and we should continue to make every effort to get the word out. DES is no longer secure for most uses, and it requires significant security expertise to evaluate those small number of cases in which it might be acceptable. Technologies exist that put DES-cracking capability within reach of a modestly financed or modestly skilled motivated attacker. There are stronger, cheaper, faster encryption algorithms available. It is time to move on.

9. Security Considerations

This entire document deals with security considerations. Still, it makes sense to summarize a few key points here. It should be clear by now that the DES algorithm offers little deterrence for a determined adversary. While it might have cost $250,000 to build a dedicated DES cracker in 1998, nowadays it can be done for considerably less. Indeed, botnets are arguably free, if you don't count the malware author's time in your cost computation.

Does this mean DES should never be used? Well, no - but it does mean that if it is used at all, it should be used with extreme care. It is important to carefully evaluate the value of the information being protected, both to its owner and to an attacker, and to fully grasp the potential risks. In some cases, DES may still provide an acceptable level of security, e.g., when you want to encrypt a file on the family PC, and there are no real threats in your household.

However, it is important to recognize that, in such cases, DES is much like a cheap suitcase lock: it usually helps honest people remain honest, but it won't stop a determined thief. Given that strong, more efficient cryptographic algorithms (e.g., AES) are available, it seems the only rational reason to continue using DES today is for compulsory backward compatibility. In such cases, if there is no plan for gradually phasing out such products, then, as a security implementer, you can do the following:

- Recommend a phased upgrade approach.
- If possible, use 3DES rather than DES (and in any case, DO NOT make DES the default algorithm!).
- Replace keys before exceeding 2^32 blocks per key (to avoid various cryptanalytic attacks).
- If there is a user interface, make users aware of the fact that the cryptography in use is not strong, and for your particular application, make appropriate recommendations in this regard.

The bottom line: it is simpler to not use this algorithm than it is to come up with narrow scenarios in which it might be okay. If you have legacy systems relying on DES, it makes sense to begin phasing them out as soon as possible.

10. Acknowledgements

The author gratefully acknowledges the contributions of Doug Whiting, Matt Curtin, Eric Rescorla, Bob Baldwin, and Yoav Nir. Their reviews, comments, and advice immeasurably improved this note. And of course, we all have the EFF and all those involved with the "Deep Crack", DESCHALL, and distributed.net efforts to thank for their pioneering research and implementations in this area.

Appendix A. What About 3DES?

It seems reasonable, given that we recommend avoiding DES, to ask: how about 3DES? Is it still safe? Thankfully, most of the discussion above does not apply to 3DES, and it is still "safe" in general. Below, we briefly explain why this is true, and what caveats currently exist.

A.1. Brute-Force Attacks on 3DES

Recall that for DES there are 2^56 possible keys, and that a brute-force attack consists of trying each key until the right one is found. Since we are equally likely to find the key on the first, second, or even last try, on average we expect to find the key after trying half (2^55) of the keys, or after 36,028,797,018,963,968 decryptions. This doesn't seem completely impossible given current processor speeds, and as we saw above, we can expect with today's technology that such an attack could almost certainly be carried out in around half a day.

For a brute-force attack on 3DES, however, the outlook is far less optimistic. Consider the problem: we know C (and possibly p), and we are trying to guess k1, k2, and k3 in the following relation:

```
C = E_k3(D_k2(E_k1(p)))
```

In order to guess the keys, we must execute something like the following (assuming k1, k2, and k3 are 64-bit values, as are Ci and p):

```
for ( k3 = 0 to 2^56 step 1 )
    compute C2 = D_k3(C1)
    for ( k2 = 0 to 2^56 step 1 )
        compute C3 = E_k2(C2)
        for ( k1 = 0 to 2^56 step 1 )
            begin
               compute p = D_k1(C3) xor IV
               if ( p equals p-expected )
                    exit loop; we found the keys
            end
```

Note that in the worst case the correct key combination will be the last one we try, meaning we will have tried 2^168 crypto operations. If we assume that each 3DES decryption (2 decryptions plus one encryption) takes a single microsecond, this would amount to 1.19 x 10^37 years. That's FAR longer than scientists currently estimate our universe to have been in existence.

While it is important to note that we could slightly prune the key space by assuming that two equal keys would never be used (i.e., k1 != k2, k2 != k3, k1 != k3), this does not result in a significant work reduction when you consider the magnitude of the numbers we're dealing with. And what if we instead assumed that technological advances allow us to apply DES far more quickly?

Today, commercial 3DES chips capable of 10-Gbps encryption are widely available, and this translates to 15,625,000 DES blocks per second. The estimate given above assumed 1,000,000 DES blocks/second, so 10-Gbps hardware is 15 times as fast. This means in the worst case it would take 7.6 x 10^35 years -- not much faster in the larger scheme of things.

Even if we consider hardware that is 1,000,000 times faster, this would still require 7.6 x 10^29 years - still FAR longer than the universe has been around. Obviously, we're getting nowhere fast here. 3DES, for all practical purposes, is probably safe from brute-force attacks for the foreseeable future.

A.2. Cryptanalytic Attacks Against 3DES

Unlike DES, there are only a few known cryptanalytic attacks against 3DES. Below, we describe those attacks that are currently discussed in the literature.

A.2.1. Meet-In-The-Middle (MITM) Attacks

The most commonly described 3DES attack is MITM, described in [HAC] and elsewhere. It works like this: take a ciphertext value 'C' (with corresponding known plaintext value 'p'), and compute the values of Cx = D_kx(C) for all possible (2^56) keys. Store each Cx,kx pair in a table indexed by Cx.

Now, compute the values of Cy = D_ki(E_kj(p)) in a nested loop, as illustrated above in our brute-force exercise. For each Cy, do a lookup on the table of Cx's. For each match found, test the triple of keys. It is important to note that a match does not imply you have the right keys - you must test this against additional ciphertext/plaintext pairs to be certain (~3 pairs for a strong measure of certainty with 3DES). Ultimately, there will be exactly one correct key triplet.

Note that computing the initial table of Cx,kx pairs requires 2^56 encryptions and 2^56 blocks of storage (about 576 gigabytes). Computing the lookup elements requires at most 2^112 cryptographic operations (table lookups are negligible by comparison), and 2^111 operations on average. Lucks [LUCKS] has come up with optimizations that reduce this to about 2^108.

3DES, even at a strength of 2^108, is still very strong. If we use our brute-force limits from above (15,625,000 blocks per second), this attack will take on the order of 6.586 x 10^17 years to carry out. Make the machine 1 million times faster, and you still need more than 658 BILLION years. We are probably safe from MITM attacks on 3DES for the foreseeable future.

A.2.2. Related Key Attacks

For a detailed description of related key attacks against 3DES (and other algorithms), see [KELSEY]. In a nutshell, for this approach the attacker knows the encryption of given plaintext under the original key K, and some related keys K'_i. There are attacks where the attacker chooses how the key is to be changed, and attacks in which the difference is known, but not controlled, by the attacker.

Here's how it works. Assume the following cryptographic relation:

    C = E_k3(D_k2(E_k1(p)))

Then, the following defines the key relation:

    K = (k1,k2,k3) and K' = (k1 + d,k2,k3)

with d being a fixed constant. Knowing p and C, we need to decrypt C under K' as follows:

    Let kx = k1 + d (note: '+' represents xor)

    and

    p' = D_kx(E_k1(p))

Once we have p', we can find kx by exhaustively trying each key until we find a match (2^56 encryptions, worst case). Once we find kx, we can conduct a double-DES MITM attack to find k2 and k3, which requires between 2^56 and 2^72 trial offline encryptions.

From a practical standpoint, it's very important to recognize the "what-if" nature of this attack: the adversary must know the plaintext/ciphertext pair, he must be able to influence a subsequent encryption key in a highly controlled fashion (or at least, know exactly how the key changes), and then have the cryptographic cooperation required to compute p'. This is clearly a very difficult attack in the real world.

A.3. 3DES Block Size

While the effective key length for 3DES is clearly much larger than for DES, the block size is, unfortunately, still only 64 bits. For CBC mode (the most commonly deployed mode in Internet security protocols), this means that, due to the birthday paradox, information about the plaintext begins to leak after around 2^32 blocks have been encrypted. For this reason, 3DES may not be the best choice for high-throughput links, or other high-density encryption applications. At minimum, care should be taken to refresh keys frequently enough to minimize ciphertext collisions in such scenarios.

Informative References

[AES] "The Advanced Encryption Standard", November 2001, <http://csrc.nist.gov/publications/fips/fips197/fips-197.pdf>.

[AES-NSA] "CNSS Policy No. 15, Fact Sheet No. 1", June 2003, <http://csrc.nist.gov/cryptval/CNSS15FS.pdf>.

[BIH93] Biham, E. and A. Shamir, "Differential Cryptanalysis of the Data Encryption Standard", 1993.

[BIH96] Biham, E., "How to Forge DES-Encrypted Messages in 2^28 Steps", 1996.

[BIH96-2] Biham, E. and A. Shamir, "A New Cryptanalytic Attack on DES", 1996.

[BLAZ96] Blaze, M., Diffie, W., Rivest, R., Schneier, B., Shimomura, T., Thompson, E., and M. Wiener, "Minimal Key Lengths for Symmetric Ciphers to Provide Adequate Commercial Security", January 1996.

[BOT05] "Know Your Enemy: Tracking Botnets", March 2005, <http://www.honeynet.org/papers/bots/>.

[CERT01] Ianelli, N. and A. Hackworth, "Botnets as a Vehicle for Online Crime", December 2005, <http://www.cert.org/archive/pdb/Botnets.pdf>.

[CURT05] Curtin, M., "Brute Force: Cracking the Data Encryption Standard", 2005.

[CURT98] Curtin, M. and J. Dolske, "A Brute Force Search of DES Keyspace", 1998, <http://www.interhack.net/pubs/des-key-crack/>.

[DES] "Data Encryption Standard", January 1977, <http://www.nist.gov>.

[DH77] Hellman, M. and W. Diffie, "Exhaustive Cryptanalysis of the NBS Data Encryption Standard", June 1977.

[DIST99] Press Release, distributed., "US GOVERNMENT'S ENCRYPTION STANDARD BROKEN IN LESS THAN A DAY", 1999, <http://www1.distributed.net/des/release-desiii.txt>.

[EFF98] EFF, "Cracking DES", July 1998.

[FBI01] "NIPC Advisory 01-003", March 2001, <http://www.fbi.gov/pressrel/pressrel01/nipc030801.htm>.

[FBI06] "FBI Webpage: Focus on Economic Espionage", January 2006, <http://www.fbi.gov/hq/ci/economic.htm>.

[FERG03] Ferguson, N. and B. Schneier, "Practical Cryptography", 2003.

[FPL02] Koeune, F., Rouvroy, G., Standaert, F., Quisquater, J., David, J., and J. Legat, "An FPGA Implementation of the Linear Cryptanalysis", FPL 2002, Volume 2438 of Lecture Notes in Computer Science, pages 846-852, Spriger-Verlag, September 2002.

[HAC] Menezes, A., van Oorschot, P., and S. Vanstone, "Handbook of Applied Cryptography", 1997.

[JAK97] Jakobsen, T. and L. Knudsen, "The Interpolation Attack on Block Ciphers", 1997.

[KELSEY] Kelsey, J., Schneier, B., and D. Wagner, "Key-Schedule Cryptanalysis of 3-WAY, IDEA, G-DES, RC4, SAFER, and Triple-DES", 1996.

[LUCKS] Lucks, S., "Attacking Triple Encryption", 1998.

[MAT93] Matsui, M., "Linear Cryptanalysis Method for DES Cipher", 1993.

[NIST-TP] "DES Transition Plan", May 2005, <http://csrc.nist.gov/cryptval/DESTranPlan.pdf>.

[NYSE1] "Extreme availability: New York Stock Exchange's new IT infrastructure puts hand-held wireless terminals in brokers' hands.", June 2005.

[RSA1] Press Release, RSA., "Team of Universities, Companies and Individual Computer Users Linked Over the Internet Crack RSA's 56-Bit DES Challenge", 1997, <http://www.rsasecurity.com/press_release.asp?doc_id=661&id=1034>.

[RSA2] Press Release, RSA., "RSA to Launch "DES Challenge II" at Data Security Conference", 1998, <http://www.rsasecurity.com/press_release.asp?doc_id=729&id=1034>.

[RSA3] Press Release, RSA., "Distributed Team Collaborates to Solve Secret-Key Challenge", 1998, <http://www.rsasecurity.com/press_release.asp?doc_id=558&id=1034>.

[RSA4] Press Release, RSA., "RSA to Launch DES Challenge III Contest at 1999 Data Security Conference", 1998, <http://www.rsasecurity.com/press_release.asp?doc_id=627&id=1034>.

[RSA5] Press Release, RSA., "RSA Code-Breaking Contest Again Won by Distributed.Net and Electronic Frontier Foundation", 1999, <http://www.rsasecurity.com/press_release.asp?doc_id=462&id=1034>.

[SCHN96] Schneier, B., "Applied Cryptography, Second Ed.", 1996.

[VA1] "Review of Issues Related to the Loss of VA Information Involving the Identities of Millions of Veterans (Report #06-02238-163)", July 2006, <http://www.va.gov/oig/51/FY2006rpts/VAOIG-06-02238-163.pdf>.

[WIEN94] Wiener, M., "Efficient DES Key Search", August 1993.

Author's Address

Scott G. Kelly
Aruba Networks
1322 Crossman Ave
Sunnyvale, CA 94089
US

EMail: scott@hyperthought.com

Acknowledgement

Funding for the RFC Editor function is currently provided by the Internet Society.