

---

# **A HACKER LOOKS AT CRYPTOGRAPHY**

**BRUCE SCHNEIER**

Counterpane Systems  
101 East Minnehaha Parkway, Minneapolis, MN 55419  
Phone: (612) 823 1098; Fax: (612) 823-1590  
schneier@counterpane.com  
<http://www.counterpane.com>

Black Hat '99  
Las Vegas, NV—7 July 1999



**COUNTERPANE SYSTEMS**

# INTRODUCTION

---

- Cryptography is a powerful tool for computers, especially networked computers.
- Cryptography allows us to take existing business and social constructs from the face-to-face world into the world of computers and networks.
- Cryptography has the potential for transforming the Internet from a toy to a serious business tool.
- Unfortunately, most products and systems that use cryptography are insecure.
- Most commercial cryptography does not perform as advertised.

# OUTLINE

---

- What are we trying to do?
- Can we do it?
- Why cryptosystems fail?
- What can we learn from this?

# PROGRAMMING SATAN'S COMPUTER

---

- Security engineering is different from any other type of engineering.
- Most products are useful for what they do.
- Security products are useful precisely because of what they do not allow to be done.
- Most engineering involves making things work.
- Security engineering involves figuring out how to make things not work...and then preventing those failures.

# PROGRAMMING SATAN'S COMPUTER (CONT.)

---

- Safety engineering involves making sure things do not fail in the presence of random faults.
- Security engineering involves making sure things do not fail in the presence of an intelligent and malicious adversary who forces faults at precisely the wrong time and in precisely the wrong way.

# TESTING SATAN'S COMPUTER

---

- Security is orthogonal to functionality.
  - Just because a security products functions properly does not mean that it's secure.
- No amount of beta testing can ever uncover a security flaw.
- Experienced security testing is required to discover security flaws.

# THE FAILURE OF TESTING SECURITY

---

- Imagine a vendor shipping a product without any functional testing.
  - No in-house testing.
  - No beta testing.
  - Just make sure it compiles and then ship it.
- A product like this will have hundreds of bugs; the odds of it working properly are negligible.
- Now imagine a vendor shipping a security product without any security testing.
- The odds of it being secure are negligible.

# HOW TO TEST SECURITY

---

- Experienced security testing can discover security flaws, but it's not easy.
- Flaws can be anywhere: the threat model, the design, the algorithms and protocols, the implementation, the configuration, the user interface, the usage procedures, and so on.
- "Black-box" testing isn't very useful.
- There is no comprehensive security checklist.
  - Experience in real-world failures is the only way to be a good tester.

# WHY CRYPTOSYSTEMS FAIL

---

- The reasons are as numerous as the number of systems.
- There are many common blunders.
- Vendors make the same mistakes over and over again.
- This list is not exhaustive, but it's pretty good.
- Vendors should feel free to make new mistakes.

# USE OF PROPRIETARY ALGORITHMS

---

- Designing cryptographic algorithms is very difficult.
  - Many published algorithms are insecure.
  - Almost all unpublished algorithms are insecure.
- Unless someone has had considerable experience cryptanalyzing algorithms, it is unlikely that his design will be secure.
  - It is easy for someone to create an algorithm that he himself cannot break.

# USE OF PROPRIETARY ALGORITHMS (CONT.)

---

- Inexperienced cryptanalysts create insecure designs (cont.):
  - If an algorithm designer has not proved that he can break published algorithms (usually by publishing his own cryptanalyses), why should anyone trust his designs?
- There is usually no reason to use an unpublished algorithm.
  - There are secure methods for making a secure, proprietary, non-interoperable algorithm out of a published algorithm.

# USE OF PROPRIETARY ALGORITHMS (CONT.)

---

- There is usually no reason to use a new and unanalyzed algorithm in place of an older and better analyzed one.
- There is no substitute for peer review.
- Never, ever, trust a proprietary or secret algorithm. (The NSA is the exception to this rule.)

# USE OF PROPRIETARY PROTOCOLS

---

- Designing cryptographic protocols is very hard.
- Many published protocols have been broken years after their publication.
- There are several protocol-design tricks that the academic community has come up with over the years.

# USE OF PROPRIETARY PROTOCOLS (CONT.)

---

- The design process of public proposal, analysis, revision, repeat seems to work pretty well.
  - Compare the security of the proprietary Microsoft PPTP with the open IPsec.
- There is no substitute for peer review.
- A closed or proprietary protocol is most likely flawed.

# BAD RANDOMIZATION

---

- Random numbers are critical for most modern cryptographic applications.
  - Session keys.
  - Seeds for generating public keys.
  - Random values for digital signatures.
  - Protocol nonces.
- An insecure random number generator can compromise the security of an entire system.
  - The security of many algorithms and protocols assumes good random numbers.

# BAD RANDOMIZATION (CONT.)

---

- There are many ways to abuse RNGs:
  - Learn the state.
  - Extend a state compromise forward or backward in time.
  - Learn or control inputs to reduce output entropy.
- Poor RNGs are probably the most common security problem in products today.
- Counterpane Systems has released Yarrow, a public-domain RNG. Use it.

# CULT OF MATHEMATICS

---

- Mathematics is a science, not a security yardstick.
- Some people obsess about key length; a long key does not equal a strong system.

# CULT OF MATHEMATICS (CONT.)

---

- Proofs of security only work within the model of security of the proof.
  - The attack on PKCS #1 went outside the mathematical model of RSA to break certain implementations.
  - One-time pads: the algorithm is provably secure, but computer applications built with the cipher are completely insecure or impractical.
- Mathematics works on bits; security involves people.

# INSECURE SOFTWARE, COMPUTERS, AND NETWORK

---

- “Buzzword compliant” cryptographic products are not sufficient.
- It is difficult to write secure code: overflow bugs, user-interface errors, difficulty of erasing secret information, etc.

# INSECURE SOFTWARE, COMPUTERS, AND NETWORK (CONT.)

---

- It can be impossible to build a secure application on top of an insecure computing platform.
  - Stealth keyboard recorders
  - Trojan horses
  - Viruses
- Insecure computer networks can undermine the security of computers attached to them.

# FAILURES OF SECURE PERIMETERS

---

- Some systems rely on the notion of a secure perimeter for security:
  - Smart cards, access tokens, dongles, etc.
- There is no such thing as tamperproof hardware.
  - The ease of defeating tamperproofing measures depends on the budget of the attacker, and changes dramatically as technology improves.

# FAILURES OF SECURE PERIMETERS (CONT.)

---

- Any system where the device is owned by one person, and the secrets within the device are owned by another, is an insecure system.
- Systems should use tamper-resistance as a barrier to entry, not as an absolute security measure.

# **NEW VULNERABILITIES INTRODUCED BY KEY ESCROW**

---

- Data backup is vital; the value of a key used to encrypt business data equals the value of that data.
- There is absolutely no business case for escrowing keys used for communication; those keys have no value.

# NEW VULNERABILITIES INTRODUCED BY KEY ESCROW (CONT.)

---

- Corporate data backup needs are not the same as law-enforcement key-escrow demands.
  - 24-hour access.
  - Surreptitious access.
  - Real-time access.
- Law-enforcement access requirements fly in the face of security requirements solved by cryptography.
  - Perfect forward secrecy.
  - Simplicity of the trust model.
  - Large targets that are very profitable to attack.

# NEW VULNERABILITIES INTRODUCED BY KEY ESCROW (CONT.)

---

- The massive scale of any key-escrow infrastructure is beyond the ability of the current security community.
- The NSA's own analysis concluded that key escrow is too risky, and creates more security problems than it solves.

# RELIANCE ON USER-REMEMBERED SECRETS

---

- Many systems rely on user-generated and user-remembered secrets for security. (Example: PGP.)
- Many password-protected systems have the characteristic of being only as secure as the weakest password.
- Users cannot remember good secrets. Period.

# RELIANCE ON USER-REMEMBERED SECRETS (CONT.)

---

- English has 1.3 bits of entropy per character; a 30-character English passphrase has as much security as a 40-bit key.
- Random passwords have less than 4 bits of entropy per character; a 12-character password is more secure than a 40-bit key.

# RELIANCE ON INTELLIGENT USERS

---

- Security works better when it is visible to the user, but users don't want to see security measures.
- Users cannot be relied on to make intelligent security decisions.
  - Users cannot be asked if they trust a piece of downloaded Java code.
  - Users cannot be expected to verify certificates.

# RELIANCE ON INTELLIGENT USERS (CONT.)

---

- Users cannot be relied on to follow security procedures.
  - Users will work around annoying security measures.
  - Users will give away secrets: social engineering.
- Cryptography works in the digital world; it is very difficult to manage the transition from people into the digital world and back again.
  - What you see is not always what you get.
  - What you get is not always what you want.

# WEAKNESSES IN AUTHENTICATION INFRASTRUCTURE

---

- Trust is a complex social phenomenon, and cannot be encapsulated in a single “certificate.”
  - There is no global name space.
  - There is no single level of assurance.
- An open, general purpose, infrastructure means more tempting targets.
  - The physical security of a CA can be enormous: US Mint–level security can be required.
  - A Trojan horse can drop a phony certificate into your computer, fooling you into trusting someone.

# WEAKNESSES IN AUTHENTICATION INFRASTRUCTURE (CONT.)

---

- Certificates issued by “trusted third parties” are useless without some sort of liability model attached.
- Key verification is not often done.
  - How may people check to see whose certificate they received when setting up an SSL connection?
  - Someone can drop a Trojan horse into your computer and fool you into trusting someone.

# WEAKNESSES IN AUTHENTICATION INFRASTRUCTURE (CONT.)

---

- Key revocation is very difficult, and is currently not being done securely.
  - When you get a key over the Internet, it is vital to verify that the key has not been revoked or stolen.
- Authentication is not the same thing as authorization.
  - Authentication is automatic; authorization requires thought.

# RELIANCE ON GLOBAL SECRETS

---

- Some systems are created with global secrets, whose compromise results in system-wide failure.
- These systems are only as secure as the most weakly protected instance of that secret.
- These systems are only as secure as the least trusted person who is trusted with that secret.
- It is very hard, sometimes impossible, to recover security after a widely deployed global secret has been compromised.

# VERSION ROLLBACK ATTACKS

---

- Real-life requirements of the Internet (and older networks) require backwards compatibility.
- For security systems, backwards compatibility is very dangerous.
  - Do you really want your secure protocol to be backwards compatible with an older, insecure, protocol?
- It is sometimes possible to convince a secure system to use an insecure version of itself.
  - Many man-in-the middle applications of this attack.

# “BELOW THE RADAR” ATTACKS

---

- Automation allows for attacks that are too cumbersome against manual systems to be profitable.
  - Marginal profitability of each attack: stealing pennies from interest-bearing accounts.
  - Marginal probability of success: spamming the net with a fraudulent business proposal.
  - Slow and boring: scanning the entire news spool looking for people who are both members of a Fundamentalist Church and active in the S&M community.

# AUTOMATED ATTACKS

---

- Many attacks are “dismissed” as requiring more skill than the average attacker has.
- Only the first attacker needs skill; the rest can use software.

# AUTOMATED ATTACKS (CONT.)

---

- Software attacks have different propagation characteristics than real attacks.
  - Physical counterfeiting is difficult; maximum damage is restricted due to the physical limitations.
  - Electronic counterfeiting can be automated; maximum damage is exponential.
- “Click here to bring down the Internet.”

# POOR FAILURE MODES

---

- Many systems have a “default to insecure” mode of operation.
  - Example: Crash a firewall. Wait until someone gets frustrated and shuts it down. Waltz right in.
  - Example: Disrupt the communications line that verifies a Visa card, and a merchant will just accept the transaction.
- Only the military has the discipline to not communicate if the security measures are not in place. (And they’re by no means perfect.)

# POOR COMPROMISE RECOVERY

---

- Systems will fail; it is essential to be able to recover from a security breach:
  - Regeneration and redistribution of compromised secrets.
  - Replacement of faulty algorithms and protocols.
- Traditionally, fraud prevention has been reactive. Automated systems can make this impossible.
- Any business plan that denies the possibility of a security compromise is unacceptable.

# IMPROPER RISK MANAGEMENT

---

- It is vital to understand the risks that a cryptographic system is protecting against.
  - Criminal attacks, publicity attacks, and legal attacks are all different. As is information warfare.
  - Privacy violations: targeted attack vs. data harvesting.
  - The characteristics—funding, expertise, access, motivation—of an attacker.

# IMPROPER RISK MANAGEMENT (CONT.)

---

- It is vital to understand the value of the data being protected.
  - Privacy, financial data, identity information, etc.
- The international nature of the Internet can exacerbate this problem:
  - Jurisdiction shopping.
- Strong systems properly manage the relative risk of the different parties being protected.
  - Systems that can leverage ongoing relationships have an easier time.
  - Anonymous systems are inherently riskier.

# POOR FORENSICS

---

- Preventing crime is much harder than detecting crime.
- Detecting crime is not enough; you need to be able to prove it in court.
- Security systems fail; audit is essential to both figure out what happened and prosecute the guilty.

# CONCLUSIONS: THE LIMITS OF PERCEPTION

---

- The problem with bad cryptography is that it looks just like good cryptography.
- Almost all security products on the market today are insecure.
- Expert security testing is the only way to tell a secure product from an insecure one.

# CONCLUSIONS: THE LIMITS OF REQUIREMENTS

---

- Cryptography doesn't have to be perfect; but the risks have to be manageable.
- Most systems permit some level of fraud.
- "A secure computer is one that has been insured."

# CONCLUSIONS: THE LIMITS OF TECHNOLOGY

---

- Cryptography is a mathematical tool. It is essential for a secure system, but it does not automatically make a system secure.
- The social problems are much harder than the mathematics.
- If you think cryptography can solve your problem, then you don't understand your problem and you don't understand cryptography.

---

**YOU ARE ALL INVITED TO  
SUBSCRIBE TO MY FREE  
MONTHLY E-MAIL NEWSLETTER:**

**CRYPTO-GRAM**

See <http://www.counterpane.com>  
for details.



**COUNTERPANE SYSTEMS**