Cryptography FAQ (04/10: Mathematical Cryptology)

Message-ID: <cryptography-faq/part04_1040202968@rtfm.mit.edu>
Supersedes: <cryptography-faq/part04_1038303482@rtfm.mit.edu>
Expires: 22 Jan 2003 09:16:08 GMT
References: <cryptography-faq/part01_1040202968@rtfm.mit.edu>
X-Last-Updated: 1994/07/05
Newsgroups:
sci.crypt,talk.politics.crypto,sci.answers,news.answers,talk.answers
Subject: Cryptography FAQ (04/10: Mathematical Cryptology)
Followup-To: poster
From: crypt-comments@math.ncsu.edu
Organization: The Crypt Cabal
Reply-To: crypt-comments@math.ncsu.edu
Date: 18 Dec 2002 09:16:16 GMT
X-Trace: 1040202976 senator-bedfellow.mit.edu 3936 18.181.0.29

Archive-name: cryptography-faq/part04
Last-modified: 93/10/10

This is the fourth of ten parts of the sci.crypt FAQ. The parts are mostly independent, but you should read the first part before the rest. We don't have the time to send out missing parts by mail, so don't ask. Notes such as ``[KAH67]'' refer to the reference list in the last part.

The sections of this FAQ are available via anonymous FTP to rtfm.mit.edu as /pub/usenet/news.answers/cryptography-faq/part[xx]. The Cryptography FAQ is posted to the newsgroups sci.crypt, talk.politics.crypto, sci.answers, and news.answers every 21 days.

Contents:

Reader, beware: This section is highly mathematical. Well, maybe not

highly mathematical, but it's got a bunch of symbols and scary-looking
formulas. You have been warned.

4.1. In mathematical terms, what is a private-key cryptosystem?

A private-key cryptosystem consists of an encryption system E and a decryption system D. The encryption system E is a collection of functions E_K, indexed by ``keys'' K, mapping some set of ``plaintexts'' P to some set of ``ciphertexts'' C. Similarly the decryption system D is a collection of functions D_K such that D_K(E_K(P)) = P for every plaintext P. That is, succesful decryption of ciphertext into plaintext is accomplished using the same key (index) as was used for the corresponding encryption of plaintext into ciphertext. Such systems, where the same key value is used to encrypt and decrypt, are also known as ``symmetric'' cryptoystems.

4.2. What is an attack?

In intuitive terms a (passive) attack on a cryptosystem is any method of starting with some information about plaintexts and their corresponding ciphertexts under some (unknown) key, and figuring out more information about the plaintexts. It's possible to state mathematically what this means. Here we go.

Fix functions F, G, and H of n variables. Fix an encryption system E, and fix a distribution of plaintexts and keys.

An attack on E using G assuming F giving H with probability p is an algorithm A with a pair f, g of inputs and one output h, such that there is probability p of computing h = H(P_1,...,P_n), if we have f = F(P_1,...,P_n) and g = G(E_K(P_1),...,E_K(P_n)). Note that this probability depends on the distribution of the vector (K,P_1,...,P_n).

The attack is trivial (or ``pointless'') if there is probability at least p of computing h = H(P_1,...,P_n) if f = F(P_1,...,P_n) and g = G(C_1,...,C_n). Here C_1,...,C_n range uniformly over the possible ciphertexts, and have no particular relation to P_1,...,P_n. In other words, an attack is trivial if it doesn't actually use the encryptions E_K(P_1),...,E_K(P_n).

An attack is called ``one-ciphertext'' if n = 1, ``two-ciphertext'' if n = 2, and so on.

4.3. What's the advantage of formulating all this mathematically?

In basic cryptology you can never prove that a cryptosystem is secure.

Read part 3: we keep saying ``a strong cryptosystem must have this property, but having this property is no guarantee that a cryptosystem is strong!''

In contrast, the purpose of mathematical cryptology is to precisely formulate and, if possible, prove the statement that a cryptosystem is strong. We say, for example, that a cryptosystem is secure against all (passive) attacks if any nontrivial attack against the system (as defined above) is too slow to be practical. If we can prove this statement then we have confidence that our cryptosystem will resist any (passive) cryptanalytic technique. If we can reduce this statement to some well-known unsolved problem then we still have confidence that the cryptosystem isn't easy to break.

Other parts of cryptology are also amenable to mathematical definition. Again the point is to explicitly identify what assumptions we're making and prove that they produce the desired results. We can figure out what it means for a particular cryptosystem to be used properly: it just means that the assumptions are valid.

The same methodology is useful for cryptanalysis too. The cryptanalyst can take advantage of incorrect assumptions. Often he can try to construct a proof of security for a system, see where the proof fails, and use these failures as the starting points for his analysis.

4.4. Why is the one-time pad secure?

By definition, the one-time pad is a cryptosystem where the plaintexts, ciphertexts, and keys are all strings (say byte strings) of some length m, and E_K(P) is just the sum (let's say the exclusive or) of K and P.

It is easy to prove mathematically that there are _no_ nontrivial single-ciphertext attacks on the one-time pad, assuming a uniform distribution of keys. Note that we don't have to assume a uniform distribution of plaintexts. (Here's the proof: Let A be an attack, i.e., an algorithm taking two inputs f, g and producing one output h, with some probability p that h = H(P) whenever f = F(P) and g = G(E_K(P)) (i.e., g = G(K + P)). Then, because the distribution of K is uniform and independent of P, the distribution of K + P must also be uniform and independent of P. But also the distribution of C is uniform and independent of P. Hence there is probability exactly p that h = H(P) whenever f = F(P) and g = G(C), over all P and C. Thus a fortiori A is trivial.)

On the other hand the one-time pad is _not_ secure if a key K is used for more than one plaintext: i.e., there are nontrivial multiple-ciphertext attacks. So to be properly used a key K must be thrown away after one encryption. The key is also called a ``pad''; this explains the name ``one-time pad.''

Also, a computer-based pseudo-random number generator does _not_ qualify as a true one-time pad because of its deterministic

properties. See `pseudo-random number generators as key stream'.

4.5. What's a ciphertext-only attack?

In the notation above, a ciphertext-only attack is one where F is constant. Given only some information G(E_K(P_1),...,E_K(P_n)) about n ciphertexts, the attack has to have some chance of producing some information H(P_1,...,P_n) about the plaintexts. The attack is trivial if it has just as good a chance of producing H(P_1,...,P_n) when given G(C_1,...,C_n) for random C_1,...,C_n.

For example, say G(C) = C, and say H(P) is the first bit of P. We can easily write down an attack---the ``guessing attack,'' which simply guesses that H(P) is 1. This attack is trivial because it doesn't use the ciphertext: it has a fifty-fifty chance of guessing correctly no matter what. On the other hand there is an attack on RSA which produces one bit of information about P, with 100% success, using C. If it is fed a random C then the success rate drops to 50%. So this is a nontrivial attack.

4.6. What's a known-plaintext attack?

The classic known-plaintext attack has F(P_1,P_2) = P_1, G(C_1,C_2) = (C_1,C_2), and H(P_1,P_2) depending only on P_2. In other words, given two ciphertexts C_1 and C_2 and one decryption P_1, the known-plaintext attack should produce information about the other decryption P_2.

Note that known-plaintext attacks are often defined in the literature as producing information about the key, but this is pointless: the cryptanalyst generally cares about the key only insofar as it lets him decrypt further messages.

4.7. What's a chosen-plaintext attack?

A chosen-plaintext attack is the first of an increasingly impractical series of _active_ attacks on a cryptosystem: attacks where the cryptanalyst feeds data to the encryptor. These attacks don't fit into our model of passive attacks explained above. Anyway, a chosen-plaintext attack lets the cryptanalyst choose a plaintext and look at the corresponding ciphertext, then repeat until he has figured out how to decrypt any message. More absurd examples of this sort of attack are the ``chosen-key attack'' and ``chosen-system attack.''

A much more important form of active attack is a message corruption attack, where the attacker tries to change the ciphertext in such a way as to make a useful change in the plaintext.

There are many easy ways to throw kinks into all of these attacks: for instance, automatically encrypting any plaintext P as

T,E_K(h(T+R+P),R,P), where T is a time-key (sequence number) chosen anew
for each message, R is a random number, and h is a one-way hash function. Here comma means concatenation and plus means exclusive-or.

4.8. In mathematical terms, what can you say about brute-force attacks?

Consider the following known-plaintext attack. We are given some plaintexts P_1,...,P_{n-1} and ciphertexts C_1,...,C_{n-1}. We're also given a ciphertext C_n. We run through every key K. When we find K such that E_K(P_i) = C_i for every i < n, we print D_K(C_n).

If n is big enough that only one key works, this attack will succeed on valid inputs all the time, while it will produce correct results only once in a blue moon for random inputs. Thus this is a nontrivial attack. Its only problem is that it is very slow if there are many possible keys.

4.9. What's a key-guessing attack? What's entropy?

Say somebody is using the one-time pad---but isn't choosing keys randomly and uniformly from all m-bit messages, as he was supposed to for our security proof. In fact say he's known to prefer keys which are English words. Then a cryptanalyst can run through all English words as possible keys. This attack will often succeed, and it's much faster than a brute-force search of the entire keyspace.

We can measure how bad a key distribution is by calculating its entropy. This number E is the number of ``real bits of information'' of the key: a cryptanalyst will typically happen across the key within
2^E guesses. E is defined as the sum of -p_K log_2 p_K, where p_K is the probability of key K.

Send corrections/additions to the FAQ Maintainer:
crypt-comments@math.ncsu.edu

Last Update January 01 2003 @ 00:33 AM