Cryptography FAQ (06/10: Public Key Cryptography)

Message-ID: <cryptography-faq/part06_1040202968@rtfm.mit.edu>
Supersedes: <cryptography-faq/part06_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 (06/10: Public Key Cryptography)
Followup-To: poster
From: crypt-comments@math.ncsu.edu
Organization: The Crypt Cabal
Reply-To: crypt-comments@math.ncsu.edu
Date: 18 Dec 2002 09:16:17 GMT
X-Trace: 1040202977 senator-bedfellow.mit.edu 3959 18.181.0.29

Archive-name: cryptography-faq/part06
Last-modified: 94/06/07

This is the sixth 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:

6.1. What is public-key cryptography?

In a classic cryptosystem, we have encryption functions E_K and decryption functions D_K such that D_K(E_K(P)) = P for any plaintext P. In a public-key cryptosystem, E_K can be easily computed from some ``public key'' X which in turn is computed from K. X is published, so that anyone can encrypt messages. If decryption D_K cannot be easily computed from public key X without knowledge of private key K, but readily with knowledge of K, then only the person who generated K can decrypt messages. That's the essence of public-key cryptography, introduced by Diffie and Hellman in 1976.

This document describes only the rudiments of public key cryptography. There is an extensive literature on security models for public-key cryptography, applications of public-key cryptography, other applications of the mathematical technology behind public-key cryptography, and so on; consult the references at the end for more refined and thorough presentations.

6.2. How does public-key cryptography solve cryptography's Catch-22?

In a classic cryptosystem, if you want your friends to be able to send secret messages to you, you have to make sure nobody other than them sees the key K. In a public-key cryptosystem, you just publish X, and you don't have to worry about spies. Hence public key cryptography `solves' one of the most vexing problems of all prior cryptography: the necessity of establishing a secure channel for the exchange of the key. To establish a secure channel one uses cryptography, but private key cryptography requires a secure channel! In resolving the dilemma, public key cryptography has been considered by many to be a `revolutionary technology,' representing a breakthrough that makes routine communication encryption practical and potentially ubiquitous.

6.3. What is the role of the `trapdoor function' in public key schemes?

Intrinsic to public key cryptography is a `trapdoor function' D_K with the properties that computation in one direction (encryption, E_K) is easy and in the other is virtually impossible (attack, determining P from encryption E_K(P) and public key X). Furthermore, it has the special property that the reversal of the computation (decryption, D_K) is again tractable if the private key K is known.

6.4. What is the role of the `session key' in public key schemes?

In virtually all public key systems, the encryption and decryption times are very lengthy compared to other block-oriented algorithms such as DES for equivalent data sizes. Therefore in most implementations of public-key systems, a temporary, random `session key' of much smaller length than the message is generated for each message and alone encrypted by the public key algorithm. The message is actually encrypted using a faster private key algorithm with the session key. At the receiver side, the session key is decrypted using the public-key algorithms and the recovered `plaintext' key is used to decrypt the message.

The session key approach blurs the distinction between `keys' and `messages' -- in the scheme, the message includes the key, and the

key itself is treated as an encryptable `message'. Under this dual-encryption approach, the overall cryptographic strength is related to the security of either the public- and private-key algorithms.

6.5. What's RSA?

RSA is a public-key cryptosystem defined by Rivest, Shamir, and Adleman. Here's a small example. See also [FTPDQ].

Plaintexts are positive integers up to 2^{512}. Keys are quadruples (p,q,e,d), with p a 256-bit prime number, q a 258-bit prime number, and d and e large numbers with (de - 1) divisible by (p-1)(q-1). We define E_K(P) = P^e mod pq, D_K(C) = C^d mod pq. All quantities are readily computed from classic and modern number theoretic algorithms (Euclid's algorithm for computing the greatest common divisor yields an algorithm for the former, and historically newly explored computational approaches to finding large `probable' primes, such as the Fermat test, provide the latter.)

Now E_K is easily computed from the pair (pq,e)---but, as far as anyone knows, there is no easy way to compute D_K from the pair (pq,e). So whoever generates K can publish (pq,e). Anyone can send a secret message to him; he is the only one who can read the messages.

6.6. Is RSA secure?

Nobody knows. An obvious attack on RSA is to factor pq into p and q. See below for comments on how fast state-of-the-art factorization algorithms run. Unfortunately nobody has the slightest idea how to prove that factorization---or any realistic problem at all, for that matter---is inherently slow. It is easy to formalize what we mean by ``RSA is/isn't strong''; but, as Hendrik W. Lenstra, Jr., says, ``Exact definitions appear to be necessary only when one wishes to prove that algorithms with certain properties do _not_ exist, and theoretical computer science is notoriously lacking in such negative results.''

Note that there may even be a `shortcut' to breaking RSA other than factoring. It is obviously sufficient but so far not provably necessary. That is, the security of the system depends on two critical assumptions: (1) factoring is required to break the system, and (2) factoring is `inherently computationally intractable', or, alternatively, `factoring is hard' and `any approach that can be used to break the system is at least as hard as factoring'.

Historically even professional cryptographers have made mistakes in estimating and depending on the intractability of various computational problems for secure cryptographic properties. For example, a system called a `Knapsack cipher' was in vogue in the literature for years until it was demonstrated that the instances typically generated could be efficiently broken, and the whole area of research fell out of favor.

6.7. What's the difference between the RSA and Diffie-Hellman schemes?

Diffie and Hellman proposed a system that requires the dynamic

exchange of keys for every sender-receiver pair (and in practice, usually every communications session, hence the term `session key'). This two-way key negotiation is useful in further complicating attacks, but requires additional communications overhead. The RSA system reduces communications overhead with the ability to have static, unchanging keys for each receiver that are `advertised' by a formal `trusted authority' (the hierarchical model) or distributed in an informal `web of trust'.

6.8. What is `authentication' and the `key-exchange problem'?

The ``key exchange problem'' involves (1) ensuring that keys are exchanged so that the sender and receiver can perform encryption and decryption, and (2) doing so in such a way that ensures an eavesdropper or outside party cannot break the code. `Authentication' adds the requirement that (3) there is some assurance to the receiver that a message was encrypted by `a given entity' and not `someone else'.

The simplest but least available method to ensure all constraints above are satisfied (successful key exchange and valid authentication) is employed by private key cryptography: exchanging the key secretly. Note that under this scheme, the problem of authentication is implicitly resolved. The assumption under the scheme is that only the sender will have the key capable of encrypting sensible messages delivered to the receiver.

While public-key cryptographic methods solve a critical aspect of the `key-exchange problem', specifically their resistance to analysis even with the presence a passive eavesdropper during exchange of keys, they do not solve all problems associated with key exchange. In particular, since the keys are considered `public knowledge,' (particularly with RSA) some other mechanism must be developed to testify to authenticity, because possession of keys alone (sufficient to encrypt intelligible messages) is no evidence of a particular unique identity of the sender.

One solution is to develop a key distribution mechanism that assures that listed keys are actually those of the given entities, sometimes called a `trusted authority'. The authority typically does not actually generate keys, but does ensure via some mechanism that the lists of keys and associated identities kept and advertised for reference by senders and receivers are `correct'. Another method relies on users to distribute and track each other's keys and trust in an informal, distributed fashion. This has been popularized as a viable alternative by the PGP software which calls the model the `web of trust'.

Under RSA, if a person wishes to send evidence of their identity in addition to an encrypted message, they simply encrypt some information with their private key called the `signature', additionally included in the message sent under the public-key encryption to the receiver. The receiver can use the RSA algorithm `in reverse' to verify that the information decrypts sensibly, such that only the given entity could have encrypted the plaintext by use of the secret key. Typically the encrypted `signature' is a `message digest' that comprises a unique

mathematical `summary' of the secret message (if the signature were static across multiple messages, once known previous receivers could use it falsely). In this way, theoretically only the sender of the message could generate their valid signature for that message, thereby authenticating it for the receiver. `Digital signatures' have many other design properties as described in Section 7.

6.9. How fast can people factor numbers?

It depends on the size of the numbers, and their form. Numbers in special forms, such as a^n - b for `small' b, are more readily factored through specialized techniques and not necessarily related to the difficulty of factoring in general. Hence a specific factoring `breakthrough' for a special number form may have no practical value or relevance to particular instances (and those generated for use in cryptographic systems are specifically `filtered' to resist such approaches.) The most important observation about factoring is that all known algorithms require an exponential amount of time in the _size_ of the number (measured in bits, log2(n) where `n' is the number). Cryptgraphic algorithms built on the difficulty of factoring generally depend on this exponential-time property. (The distinction of `exponential' vs. `polynomial time' algorithms, or NP vs. P, is a major area of active computational research, with insights very closely intertwined with cryptographic security.)

In October 1992 Arjen Lenstra and Dan Bernstein factored 2^523 - 1 into primes, using about three weeks of MasPar time. (The MasPar is a 16384-processor SIMD machine; each processor can add about 200000 integers per second.) The algorithm there is called the ``number field sieve''; it is quite a bit faster for special numbers like 2^523 - 1 than for general numbers n, but it takes time only exp(O(log^{1/3} n log^{2/3} log n)) in any case.

An older and more popular method for smaller numbers is the ``multiple polynomial quadratic sieve'', which takes time exp(O(log^{1/2} n log^{1/2} log n))---faster than the number field sieve for small n, but slower for large n. The breakeven point is somewhere between 100 and 150 digits, depending on the implementations.

Factorization is a fast-moving field---the state of the art just a few years ago was nowhere near as good as it is now. If no new methods are developed, then 2048-bit RSA keys will always be safe from factorization, but one can't predict the future. (Before the number field sieve was found, many people conjectured that the quadratic sieve was asymptotically as fast as any factoring method could be.)

6.10. What about other public-key cryptosystems?

We've talked about RSA because it's well known and easy to describe. But there are lots of other public-key systems around, many of which are faster than RSA or depend on problems more widely believed to be difficult. This has been just a brief introduction; if you really want to learn about the many facets of public-key cryptography, consult the books and journal articles listed in part 10.

6.11. What is the ``RSA Factoring Challenge''?

[Note: The e-mail addresses below have been reported as invalid.]
In ~1992 the RSA Data Securities Inc., owner and licensor of multiple patents on the RSA hardware and public key cryptographic techniques in general, and maker of various software encryption packages and libraries, announced on sci.math and elsewhere the creation of an ongoing Factoring Challenge contest to gauge the state of the art in factoring technology. Every month a series of numbers are posted and monetary awards are given to the first respondent to break them into factors. Very significant hardware resources are required to succeed by beating other participants. Information can be obtained via automated reply from

challenge-rsa-honor-roll@rsa.com
challenge-partition-honor-roll@rsa.com

Part01 - Part02 - Part03 - Part04 - Part05 - Part06 - Part07 - Part08 - Part09 - Part10

[By Archive-name | By Author | By Category | By Newsgroup]
[Home | Latest Updates | Archive Stats | Search | Usenet References | Help]

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

Last Update January 01 2003 @ 00:33 AM