

Applications of Factoring and
Discrete Logarithms to Cryptography
or

The Invention of Public Key Cryptography

Sam Wagstaff

Computer Sciences and Mathematics

October 21, 2008

Cryptography before the 1970s

Cryptography has been used to hide messages at least since the time of Julius Caesar more than 200 years ago.

Usually, the sender and receiver would meet some time before the secret communication and decide how to hide their future secret message. They would choose an algorithm, which remained fixed for a long time, and a key, which might change periodically. Or, perhaps a trusted messenger would carry a new key from one to the other.

We review old-fashioned cryptography, most of it from before the 1970s.

Junior Space Cadet to AES.

NSA was created by President Truman in 1952 to protect US secret communication and to attack that of other countries. It performed this mission secretly and apparently with great success until at least the 1970s. It believed that it should control all work on cryptography done in the United States.

Some businesses needed cryptography to communicate secretly with their offices overseas.

On March 17, 1975, a Federal Register posting by the national Bureau of Standards announced a cipher, DES, approved for individuals and businesses.

It was soon suspected that NSA had approved it after putting secret weaknesses into it (S-boxes) that would allow it, but not others, to break it easily.

In the early 1970s, people began to hook computers together and think about email and other forms of electronic communication.

Computer theoreticians began to ponder how one could “sign” an electronic document so that the reader could be certain who wrote it. If each person used a unique long number as a signature, say, it could be copied perfectly by a forger.

Others pondered how to create a system where people who had never met could communicate securely. Recall that one-key ciphers require users to meet and exchange keys before their secret communication. All ciphers known then were of this type.

Whit Diffie and Marty Hellman thought about these matters and came up with several brilliant solutions in a 1976 paper.

First, they considered one-way functions, easy to compute (forwards), but almost impossible to invert. A function f is one-way if it is easy to compute $f(x)$ for any x , but, given almost any value y in the range, it is hard to find any x with $f(x) = y$.

Computer password files were under attack in the early 1970s. George Purdy and others suggested storing $f(\text{password})$ in a file rather than the plaintext password. The login program would compute f of whatever password you type and compare it with the password file entry.

Example: a high-degree, sparse polynomial modulo p , like $f(x) = (x^{864391} + 43937x^{1023} + 461x^{17} + 23) \bmod p$.

Diffie and Hellman invented a method of key exchange based on the one-way function

$$f(x) = b^x \bmod p.$$

Given b , p and y , it is hard to find a “discrete logarithm” x with $f(x) = y$. It is easy to compute $b^x \bmod p$ by “fast exponentiation.”

Fast Exponentiation

Input: An integer $x \geq 0$, a prime p and a number b .

Output: The value $y = b^x \bmod p$.

```
e = x
y = 1
z = b mod p
while (e>0) {
    if (e is odd) y = y * z mod p
    z = z * z mod p
    e = e/2
}
return y
```

The Diffie-Hellman Key-Exchange Protocol

This protocol allows two users to choose a common secret key, while communicating over a channel with eavesdroppers.

The two users agree on a common large prime p and a constant value g , which may be publicly known and available to everyone.

Alice secretly chooses a random A in $0 < A < p - 1$ and computes $y_A = g^A \bmod p$. She sends y_A to Bob. Bob secretly chooses a random B in $0 < B < p - 1$ and computes $y_B = g^B \bmod p$. He sends y_B to Alice.

Alice computes $K_A = y_B^A \bmod p$. Bob computes $K_B = y_A^B \bmod p$. Now

$$K_B \equiv y_A^B \equiv g^{A \cdot B} \equiv y_B^A \equiv K_A \bmod p,$$

so both have chosen the same key $K_A = K_B$.

Knowing p , g , y_A and y_B is not enough information to compute $K_A = K_B$, A or B .

This key-exchange protocol provides secure communication between one user and whomever is at the other end of the connection. It is also subject to the man-in-the-middle attack.

The second innovation in the Diffie-Hellman paper of 1976 solved this problem. Their idea was to split the key!

Bob would use a cipher with two independent keys. He would make public his enciphering key and the enciphering algorithm. His deciphering algorithm would be public, but only Bob would know his deciphering key.

To send Bob a secret message, Alice would find his public key and use it to encipher the message. Once enciphered, no one but Bob could decipher it.

A public-key cipher lacks authenticity. Anyone could write a letter to Bob, sign it “Alice,” encipher it using Bob’s public key, and send it to Bob. Bob would not know whether it came from Alice.

The third innovation of Diffie and Hellman was the notion of a digital signature.

Alice could construct her own public and private keys and “sign” a message by applying the deciphering algorithm with her private key to the plaintext message.

A recipient of this signed message would be certain that it came from Alice when he obtained her public key, used it with the enciphering algorithm, and obtained meaningful text from Alice. Only Alice could have constructed a message with this property because only Alice knows her private key.

Diffie and Hellman did not propose enciphering and deciphering algorithms for public-key cryptography in their paper.

Ron Rivest, Adi Shamir and Leonard Adleman read their paper and tried many complicated algorithms for doing this. One or two of them would propose a scheme and the other would break it. Some of these schemes involved the difficulty of factoring large integers.

On April 3, 1977, Rivest discovered the system now called RSA. It stripped down the idea of factoring to the bare essentials. He gave a simple formula for enciphering a message. It used the product of two large primes, which would be public. He also gave a formula for deciphering the ciphertext which used the public information and information derived from the two large primes.

The RSA public-key cipher

Let $n = pq$ be the product of two large primes. Choose a random e prime to $\phi(n) = \phi(pq) = (p-1)(q-1)$. Use the Euclidean algorithm to compute d with $ed \equiv 1 \pmod{(p-1)(q-1)}$.

Encode plaintext as (blocks) $0 \leq M < n$. Encipher M as $C = E(M) = M^e \pmod n$. Decipher C as $M = D(C) = C^d \pmod n$.

This works, that is, $D(E(M)) = M$ for all M in $0 \leq M < n$, provided that $ed \equiv 1 \pmod{\phi(n)}$, since $M^{\phi(n)} \equiv 1 \pmod n$ by Euler's Theorem. (Proof: Write $ed = t\phi(n) + 1$ for some integer t .)

Each user of RSA has her own set of keys: Make n and e public, but keep d secret. The factors p and q are not needed after e and d are computed, but in any case should not be revealed.

The first version of the RSA paper was MIT Tech Memo 18, “A Method for Obtaining Digital Signatures and Public Key Cryptosystems,” April 4, 1977.

Martin Gardner wrote a column in the August, 1977, *Scientific American* describing the work of Diffie, Hellman, Rivest, Shamir and Adleman. In it, RSA offered a challenge message encoded with a 129-digit modulus and $e = 9007$. The modulus was factored in 1993–1994 by Derek Atkins, Michael Graff, Arjen Lenstra and Paul Leyland who supervised 600 volunteers using 1600 machines.

In the column, RSA offered copies of their Tech Memo for a SASE. Thousands requested it. NSA blocked its mailing and tried to prevent publication, citing the ITARs. It was aghast at the Diffie-Hellman paper and then the RSA discovery. NSA would not allow NSF to fund cryptography.

Senator Frank Church's Intelligence Committee was investigating NSA's eavesdropping practices. Eventually, NSA's new director, Vice Admiral Bobby Inman backed down and allowed publication. NSF agreed to give NSA copies of crypto proposals.

In December, 1977, R, S and A invited students to a pizza and envelope-stuffing party. Thousands of copies of the paper were mailed out. It introduced "Alice" and "Bob" to the crypto world.

It was revealed in 1997 that James Ellis of GCHQ invented public key crypto (like Diffie-Hellman) in 1969 and Cliff Cocks of GCHQ invented the RSA cipher in 1973.

After the RSA paper appeared, others began inventing new public key cryptosystems. Many schemes, like knapsacks, were quickly shot down.

At CRYPTO '84, Taher ElGamal announced a public key cryptosystems based on the difficulty of the discrete logarithm problem.

DLP for a multiplicative group: Given group elements P and Q find x so that $Q = P^x$. (x is sort of $\log_P Q$.)

DLP for an additive group: Given group elements P and Q find x so that $Q = x \cdot P$. (x is sort of Q/P , but it is still called a discrete logarithm.)

Here is the ElGamal public key cipher for an additive group. (ElGamal did it first for a subgroup of the the multiplicative group of integers modulo a prime.)

Public are a large finite cyclic group G (of prime order p) and a generator $g \in G$. Assume it is easy to choose random group elements and to add two group elements quickly. Also, there is a fast way to encode a message as a group element and to extract a message from any group element.

Alice begins by choosing a secret integer $0 < a < p$. She computes $P_A = a \cdot g$. She makes P_A public and remembers the secret a .

Bob sends a message to Alice by encoding it in an element $P \in G$. He chooses a random integer $0 < x < p$. He obtains Alice's public key P_A . He enciphers the message as the pair of group elements $(x \cdot g, (x \cdot P_A) + P)$, which he sends to Alice.

An eavesdropper who sees the pair $(x \cdot g, (x \cdot P_A) + P)$ in transit cannot obtain the message P because it is hidden by having $x \cdot P_A$ added to it, and he doesn't know x . He knows g and could get x if he could solve the DLP $x \cdot g =$ the first component of the pair. But the DLP is hard and the ciphertext looks like a pair of random group elements.

Say Alice receives the message as the pair (y, t) . She recalls her secret a and computes

$$a \cdot y = (ax \cdot g) = x \cdot (a \cdot g) = x \cdot P_A.$$

Then the plaintext is

$$t - a \cdot y = (x \cdot P_A) + P - (x \cdot P_A) = P.$$

The second component holds the message disguised by having $x \cdot P_A$ added to it. The first component is a hint for computing $x \cdot P_A$, but one useful only to someone who knows the secret a , and Alice is the only person who knows a .

It may be possible to break RSA without factoring the public modulus or ElGamal without solving a DLP, but it appears that if the keys are chosen properly, the only way to break the cipher is to solve the hard problem.

How hard is factoring the product $n = pq$ of nearly equal primes?

Trial division takes $O(\sqrt{n})$ steps. This is too slow, but sometimes cited in popular articles: “It would take 10^{80} years to factor a number of this size.”

The quadratic sieve factors n in time

$$L(n) = \exp\left(\sqrt{(\log n)(\log \log n)}\right).$$

For all $\epsilon > 0$ and all K we have

$$(\log n)^K < L(n) < n^\epsilon$$

for all sufficiently large n .

The basic idea of QS (and CFRAC and NFS) is that you have a good chance to factor n if you can find two integers x and y so that $x^2 \equiv y^2 \pmod{n}$. Then n divides

$$x^2 - y^2 = (x + y)(x - y).$$

If $x \not\equiv \pm y \pmod{n}$, then $\gcd(n, x + y)$ and $\gcd(n, x - y)$ are non-trivial factors of n .

When n has just two prime factors, we are done if we find one of them.

It is not fruitful to compute

$$1^2, 2^2, 3^2, 4^2, \dots \pmod{n}$$

and hope for a repeated value. This would take $O(\sqrt{n})$ steps.

The QS forms many squares $f(x) = x^2 \bmod n$, with x chosen to make $f(x)$ small, factors the numbers $f(x)$ and matches their prime factors to form a square.

Example: Let's factor $n = 1649$.

$$f(41) = 41^2 \bmod n = 1681 \bmod n = 32 = 2^5$$

$$f(42) = 42^2 \bmod n = 1764 \bmod n = 115 = 5 \cdot 23$$

$$f(43) = 43^2 \bmod n = 1849 \bmod n = 200 = 2^3 \cdot 5^2$$

$$f(44) = 44^2 \bmod n = 1936 \bmod n = 287 = 7 \cdot 41$$

Note that $32 \cdot 200 = 2^8 \cdot 5^2 = 80^2$, so

$$(41 \cdot 43)^2 \equiv 80^2 \bmod 1649.$$

Also, $41 \cdot 43 = 1763 \equiv 114 \bmod 1649$ and we have $\gcd(114 - 80, 1649) = 17$ and $\gcd(114 + 80, 1649) = 97$, so $1649 = 17 \cdot 97$.

The auxiliary numbers factored by the QS can be factored efficiently by a “sieve.” Another key ingredient in the algorithm is the fact that a positive fraction of the auxiliary numbers have all of their prime factors less than a fixed upper bound, so there will be plenty of them.

The Number Field Sieve, the current fastest known integer factoring algorithm does the same matching of prime factors in millions of identities of the form:

$$\text{small integer} \equiv \text{small algebraic integer} \pmod{n}$$

All the integers and the algebraic integers are factored efficiently by sieves. Linear algebra over $GF(2)$ is used to match their prime factors on both sides and form congruent squares. The final step is the same as for QS. The time complexity to factor n by NFS is

$$\exp\left(2(\log n)^{1/3}(\log \log n)^{2/3}\right).$$

In a general group of order p , the DLP can be solved in $O(\sqrt{p} \log p)$ steps, and this is the fastest known time. Several methods achieve this speed.

Dan Shanks' Baby-step-giant-step method solves $Q = x \cdot P$ as follows.

Let $m = \lceil \sqrt{p} \rceil$.

Compute and sort the m pairs (j, mjP) , for j from 0 to $m - 1$, by second component.

Compute and sort the m pairs $(i, Q - iP)$, for i from 0 to $m - 1$, by second component.

Find a pair (j, y) in the first list with the same second component as a pair (i, y) in the second list. The search will succeed because every integer between 0 and $p - 1$ can be written as a two-digit number ji in base m .

Finally, $x = (mj + i) \bmod p$.

The DLP $P^x \equiv Q \pmod{p}$ may be solved by much faster methods similar to QS or NFS. The QS analogue is called the index calculus method and finds x in time $L(p)$.

Similar fast methods also apply to supersingular elliptic curves and perhaps some elliptic curves with complex multiplication. This is the subject of current research.

Any elliptic curve can be embedded into a finite field, with point addition corresponding to multiplication of field elements. Usually, the finite field is enormous. But in certain cases (supersingular), the finite field is about as large as the elliptic curve. In these cases the index calculus solves the DLP quickly.