10th International Research/Expert Conference
”Trends in the Development of Machinery and Associated Technology”
TMT 2006, Barcelona-Lloret de Mar, Spain, 11-15 September, 2006

DIGITAL SIGNATURE ALGORITHM (DSA)

Assist. Hermina Alajbegović, B. Bc.
Faculty of Mechanical Engineering
Zenica B&H
hermina@mf.unze.ba

Ph.D. Hasan Jamak
University of Sarajevo
Faculty of natural sciences and mathematics,

Ph.D. Dževad Zečić
Faculty of Mechanical Engineering
Zenica
B&H

ABSTRACT

This paper presents one of the cryptography solutions about establishing user's identity and authentic documents on the Internet which are exchanged among users. These problems are very prevalent because the Internet plays an important and dominant role in the business world. In this paper, the existing solutions will be brought closer the readers who are not very familiar with cryptography and algebra. Codes of these solutions are created in Mathematica 4.0.

Key words: computer security, digital signature, public-key cryptography, Federal Information Processing Standard.

1. INTRODUCTION

In today’s developing and competitive business world, a great number of banks and other institutions use the Internet for servicing their customers. The exchange of documents requires special security measures to safeguard the information from unauthorized access. A digital signature is an electronic signature that can be used to authenticate the identity of the sender of a message or the signer of a document, and possibly to ensure that the original content of the message or document that has been sent is unchanged. Digital signatures are easily transportable, cannot be imitated by someone else, and can be automatically time-stamped (date and time are added to message and then that message is signed). The ability to ensure that the original signed message arrived means that the sender cannot easily repudiate it later. A digital signature can be used with any kind of message, whether it is encrypted or not, simply so that the receiver can be sure of the sender's identity and that the message arrived intact. Our primary goal is to simplify the application of the existing algorithms for this problem and in the continuation we provide those solutions.

2. DSA

There is a vast number of digital signature algorithms. A digital signature method generally defines two complementary algorithms; one for signing and the other for verification, while the output of the signing process is called a digital signature. In this paper, we explain some consideration of The Digital Signature Algorithm (DSA). That is a United States Federal Government standard for digital signatures. It was proposed by the National Institute of Standards and Technology (NIST) in August 1991 for use in their Digital Signature Standard (DSS), specified in FIPS 186 [1], adopted in 1993. The DSA digital signature is a pair of large numbers represented in a computer as strings of binary digits. The digital signature is computed using a set of rules (i.e., the DSA) and a set of parameters such that the identity of the signatory and integrity of the data can be verified. Security of DSA depends on the intractability of the discrete logarithm problem: Find x, $0 \le x \le p-2$, such that $\alpha^x \equiv \beta(\mathrm{mod}\, p)$, where is given (known) a prime p, a generator α of $\mathbb{Z}_p$, and an element β of $\mathbb{Z}_p$. Note: sign "$\equiv$" marks relation equivalence which is defined in following way: $x \equiv y$ if only if $x - y = np$ for some integer number n (i.e.$n \in \mathbb{Z}$). Set of all classes of equivalence of that relation $\mathbb{Z}_p$ is set of all possible remainders of the division of some number by p $\left(\text{i.e.}\mathbb{Z}_p = \left\{\bar{0}, \bar{1}, \ldots \left(\overline{p-1}\right)\right\}\right)$. $\mathbb{Z}_p$ is a cyclic group, so it has a generator. Note: group G is cyclic if it is generated entirely by some $g \in G$, i.e. every $a \in G$ is equal to g^k for some $k \in \mathbb{Z}$.

The Secure Hash Algorithm (SHA-1) is used in the signature generation process to obtain a condensed version of data, called a message digest. For a message of length less than 2^{64} bits, the Secure Hash Algorithm (SHA-1) produces a 160-bit condensed representation of the message called a message digest. The message digest is used during generation of a signature for the message. The SHA-1 is also used to compute a message digest for the received version of the message during the process of verifying the signature. Any change to the message in transit will, with very high probability, result in a different message digest, and the signature will fail to verify.
You can find more detailed information about this algorithm in references [1] or [3]. DSA is not a very complex algorithm, but we could be generate public key and digital signature, we need array other algorithms: algorithm Miller-Rabin probabilistic primality test (to check whether number is prime), algorithm for generating prime numbers, algorithm for computing multiplicative inverses in $\mathbb{Z}_n$, extended Euclidean algorithm, secure hash algorithm. This problem becomes complex because we need to deal with large numbers. If we were to use Pascal for creating the code for these algorithms, we would encounter serious problems, because Pascal cannot deal with large number and we would need to create a calculator for the big numbers. This calculator would consist of approximately twenty complex procedures (addition of two numbers, subtraction of two numbers, division of two numbers, product of two numbers, conversion of numbers to binary form, conversion to hexadecimal form, computing multiplicative inverses in $\mathbb{Z}_n$, test_signed_number, convert_number_in_long_number, etc.) and about ten functions (check the equality of two number, strictly greater, strictly less, etc.). Large numbers are defined by means of records (see Fig.1.).

```
const
  max_length=1000;
type
 Unsigned_long_number = record
                                date:   array[0..   max_length]   of
integer;
                                start: integer;
                                finish: integer;
                             end;
  long_number = record
                 magnitude : unsigned_long_number;

                  is_negative: boolean;

                 end;
```

Fig. 1. Definition of a long number in Pascal

The complete code in Pascal would consist of twenty-seven pages. The code for the above mentioned calculator alone is approximately twenty pages.
As an illustration, we give here one procedure from the code of calculator, which is required for DSA in Pascal. This procedure returns (as a result) quotient of two unsigned number (see Fig2.)

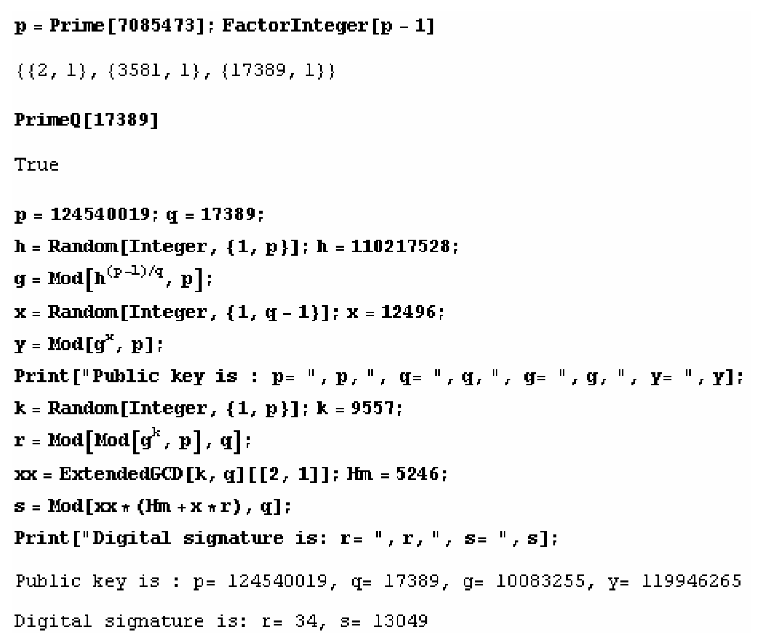
```
procedure unsigned_long_division (a,b: unsigned_dugi_broj;
                                   var quotient_unsigned_long_number;
                                 var remainder : unsigned_long_number);
   var
     temp1, temp2, temp3, temp4, one :unsigned_dugi_broj;
   begin
     unsigned_normalize(a);
     unsigned_normalize(b);
     set_as_zero(quotient);
     set_as_one(one);
    If ((b.finish=0) and (b.data[0]=0)) then
     begin
        writeln ('Division by zero attempted');
        exit;
     end;
      while (unsigned_greater_or_equal(a,b)) do
                   begin
                        a.start:=a.start + 1;
                    end;
               unsigned_shift(b, a.start – 1, temp1);
               unsigned_shift(one, a.start – 1, temp2);
               a.start:=0;
               unsigned_difference(a, temp1, temp3);
               a:=temp3;
               unsigned_sum(quotient, temp2, temp4);
               quotient:=temp4;
        end;
     remainder:=a;
   end;
```

Fig. 2. code of procedure unsigned long division in Pascal

The running time for this complex and lengthy program is rather long and unacceptable, so we must try to find a more suitable and efficient solution.

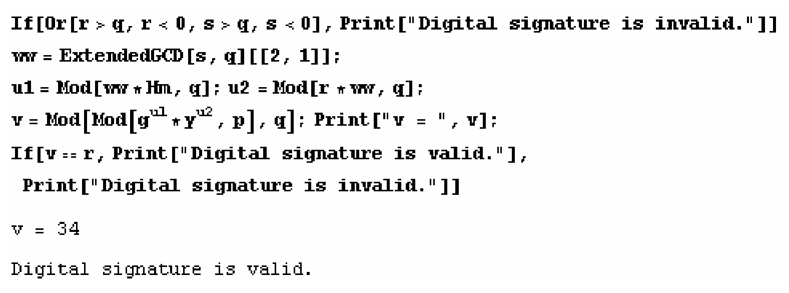
3. DSA IN MATHEMATICA 4.0

Mathematica 4.0 is software that has an immense computational power. In standard package contains the world's largest collection of computational algorithms. The package includes the algorithm for primality test and extended Euclidean algorithm.
Command PrimeQ[expr] yields True if *expr* is a prime number, and yields False otherwise. FactorInteger[*n*] gives a list of the prime factors of the integer *n*, together with their exponents. Therefore, it is not necessary to create these algorithms for the DSA. We simply use the existing algorithms included in Mathematica 4.0.
Now, we will demonstrate generation and verification signature with DSA in Mathematica 4.0. First, if we want create digital signature, we have to create the public key. The public key consists of four numbers (*p*, *q*, *x*, *y*). *q* is a prime number such that $2^{159} < q < 2^{160}$, *p* is a prime number with the property that q divides $(p-1)$ and $2^{L-1} < p < 2^{L}$ for $512 \leq L \leq 1024$ and *L* a multiple. The private key is *x* and *k*. Those numbers have to be secret, because security of DSA depends on maintaining the secrecy of the private key. We have to choose number *k* again for every new signature.
We give an example of the code in Mathematica required for generating the public key and generating the digital signature (see ig3. and fig 4.). We choose concrete numbers for *h, x, k* and message digest (*Hm*) for testing.



```
p = Prime[7085473]; FactorInteger[p - 1]

{{2, 1}, {3581, 1}, {17389, 1}}

PrimeQ[17389]

True

p = 124540019; q = 17389;
h = Random[Integer, {1, p}]; h = 110217528;
g = Mod[h^((p-1)/q), p];
x = Random[Integer, {1, q - 1}]; x = 12496;
y = Mod[g^x, p];
Print["Public key is : p= ", p, ", q= ", q, ", g= ", g, ", y= ", y];
k = Random[Integer, {1, p}]; k = 9557;
r = Mod[Mod[g^k, p], q];
xx = ExtendedGCD[k, q][[2, 1]]; Hm = 5246;
s = Mod[xx * (Hm + x * r), q];
Print["Digital signature is: r= ", r, ", s= ", s];

Public key is : p= 124540019, q= 17389, g= 10083255, y= 119946265
Digital signature is: r= 34, s= 13049
```

Fig. 3 Code of key generation for the DSA and signature generation in Mathematica 4.0



```
If[Or[r > q, r < 0, s > q, s < 0], Print["Digital signature is invalid."]]
ww = ExtendedGCD[s, q][[2, 1]];
u1 = Mod[ww * Hm, q]; u2 = Mod[r * ww, q];
v = Mod[Mod[g^u1 * y^u2, p], q]; Print["v = ", v];
If[v == r, Print["Digital signature is valid."],
 Print["Digital signature is invalid."]]

v = 34

Digital signature is valid.
```

Fig. 4 DSA signature verification in Mathematica 4.0

The complete code for the DSA algorithm, including the code for SHA, consists of two pages. In Mathematica 4.0, command Timing[*expr*] evaluates *expr*, and returns a list of time used, together with the result obtained. The run time for the code of DSA is about half a second in Mathematica 4.0.

4. CONCLUSION

We can use the DSA to generate the digital signature for all kinds of message (encrypted or not). This algorithm has proven to be very safe, because it depends of discrete logarithm problem, which is very difficult for solve. The code for the DSA algorithm can be generated using many different programming languages, especially using program applications in mathematical programming, such as Mathematica. We can also get the wanted results by using Pascal, but the code is very complex and the running time is very long. With the help of Mathematica we can generate the digital signature simply and very quickly.

5. REFERENCE

[1] A. Menezes, P. van Oorschot, I S. Vanstone, “Handbook of Applied Cryptography”, CRC Press, Inc., 1996 god.

[2] Bruce Schneier, “Applied cryptography”, John Wiley & Son, Inc. ,USA, second edition 1996.

[3] Web page www.FIPS 186 –(DSS), Digital Signature Standard

[4] Web page www.itl.nist.gov/**fips**pubs/fip180-1.htm, Secure Hash Standard