Secure Network Bootstrapping:
An Algorithm for Authentic Key Exchange and Digital Signatures

Josep Domingo-Ferrer and Llorenç Huguet-Rotger

Departament d'Informàtica, Universitat Autònoma de Barcelona, Bellaterra (Barcelona)
Catalonia, Spain

A new method is presented which enables key exchange without any previous secret agreement; a variant of it yields a signature scheme. This method requires less interactive computation than usual public key protocols, and also provides secrecy and authentication, so that its implementation is very attractive from a cost point of view.

Keywords: Network security, Key exchange, Digital signatures, Authentication protocols, Cryptography, Computer networks.

1. Introduction

A new key exchange algorithm based on the discrete logarithm is presented. Protection against impersonation (vs. Diffie-Hellman exponential key exchange) and the small amount of interactive computation (vs. public key RSA-like methods), when using precalculation and parallel processing, figure among its main features.

A variant of the key exchange algorithm yields a signature scheme. In this case, the advantage is that shorter signatures with a faster verification (vs. other logarithm-based signatures, e.g. ElGamal) are obtained.

Section 2 introduces some elementary mathematical concepts that will be used throughout the paper. Section 3 specifies the necessary assumptions and initializations for the method proposed here. Section 4 describes the key exchange protocol. Section 5 presents a variation of the key exchange algorithm, which yields the signature scheme. In Section 6, a number of attacks to the key exchange and signature schemes are outlined and countered. Section 7 reviews two current key exchange algorithms (Diffie-Hellman and RSA) and a signature scheme (ElGamal), and points out their weaknesses and disadvantages. Finally, Section 8 contains a few concluding remarks.
2. Mathematical background

In order to improve the readability of this paper, we will remember here two mathematical tools that will appear in the rest of the paper, namely the Galois fields and the discrete logarithm problem. The reader familiar with both of these matters may skip to the next section.

Let $q$ be an integer. Now given the addition mod $q$, the multiplication mod $q$, and a set $Z/(q)$ of integers mod $q$, i.e. 0 through $q-1$, we have a finite or Galois field if every element $a$ of $Z/(q)$ has its multiplicative inverse $a^{-1}$ in $Z/(q)$, such that $a \cdot a^{-1}$ mod $q = 1$. In particular, if $q$ is chosen to be a prime, an elementary algebraic result tells us that this last property holds and thus $Z/(q)$ is always a Galois field when $q$ is a prime. It is clear that addition mod $q$ and multiplication mod $q$ are both closed operations in $Z/(q)$, so that they can be carried out in the usual way except for a final mod $q$ computation.

A primitive element $\alpha$ of a Galois field $Z/(q)$ is one whose powers generate all the non-zero elements $1, 2, ..., q - 1$ of the field; it can be shown that for every Galois field there exists a primitive element. Now if

$$Y = \alpha^x \mod q$$

for $1 < Y < q - 1$, then $X$ is called the logarithm of $Y$ to the base $\alpha$, over $Z/(q)$, that is

$$X = \log_{\alpha} Y$$

over $Z/(q)$, for $1 < Y < q - 1$. Calculation of $Y$ from $X$ is easy; for example, by using repeated squaring, it takes at most $2 \cdot \log_2 q$ multiplications. For instance,

$$\alpha^{21} = \alpha^{16} \cdot \alpha^{4} \cdot \alpha = [(\alpha^2)^2]^2 \cdot (\alpha^2)^2 \cdot \alpha$$

Computing $X$ from $Y$ is usually much more difficult, and it is known as the discrete logarithm problem (see for example ref. [1]). If $q$ has been chosen suitably, extracting logarithms mod $q$ needs a precomputation proportional to

$$L(q) = e^{(\ln q \cdot \ln q)^{1/2}}$$

After this precomputation individual logarithms can however be found rather quickly.

3. Initialization

Let us define the necessary elements for our system.

$q$ is a large prime (say 150 digits) and $\alpha$ a primitive element of $Z/(q)$, both publicly known; $q - 1$ must have at least one large prime factor. Each node $i$ in the network has a very probably unique name (later we shall see what this means) in the form $ID_i = \alpha^{t_i} \mod q$, where $t_i$ is randomly chosen by node $i$ in the range $2,...,q - 1$ and remains only known to node $i$. $ID_i$ is public for all $i$ in the network, and together with $\alpha$ and $q$ is available on all communication media, in order to avoid substitutions.

To perform initialization, a particular node in the network (only one) generates $\alpha$ and $q$ and broadcasts them by all available means. After this, each node $i$ in the network

(1) Generates a number $t_i$ with $2 \leq t_i \leq q - 1$ and keeps it secret.

(2) Computes $ID_i = \alpha^{t_i} \mod q$ and also broadcasts $ID_i$.

Now every node knows $\alpha$, and each other's name. The probability for all nodes to have different names is

$$P(q - 3, n)$$

$$PR(q - 3, n)$$

where $n$ stands for the number of nodes, $P$ denotes permutations and $PR$ permutations with repetition. It is straightforward to see that this quantity approaches unity if $q \geq n^2$. 146
4. A Key Exchange Protocol

With the previous assumptions, we present a key exchange protocol providing secrecy and authentication. From now on, we shall reduce to an exchange between two nodes $i$ and $j$, which performs the following steps:

**STEP 1. NODE j:** Compute $X_1$, such that
\[ \text{gcd}(X_1, q-1) = 1 \text{ and } 2 \leq X_1 \leq q-2 \] (1)
Compute also $Y_1 = \alpha^{X_1} \mod q$
Send $Y_1$ and $\text{ID}_j$ to NODE $i$

**STEP 2. NODE i:** Upon receiving $Y_1$, pick $X_2$ and $X_4$ such that
\[ X_2 + X_4 = ti \mod (q-1) \text{ and } 2 \leq X_2, X_4 \leq q-2 \] (2)
Compute the key $K = \alpha^{X_2} \mod q$
Compute $Y_2 = Y_1^{X_2} \mod q$
Compute $Y_4 = Y_1^{X_4} \mod q$
Send $Y_2$ and $Y_4$ to NODE $j$

**STEP 3. NODE j:** Now NODE $j$ computes the multiplicative inverse of $X_1 \mod (q-1)$. $X_1^{-1}$ such that
\[ X_1^{-1} \cdot X_1 = 1 \mod (q-1) \] (3)
By computing
\[ Y_2^{X_1^{-1}} \mod q = K \]
NODE $j$ retrieves the key $K$ it will share with NODE $i$ from now on. An additional exponentiation yields
\[ Y_4^{X_1^{-1}} \mod q = \alpha^{X_4} \mod q \] (4)

In order to achieve authentication, NODE $j$ performs the following test on $K$:
\[ K \cdot \alpha^{X_4} \mod q = \text{ID}_i = \alpha^i \mod q \] (5)
If eqn. (5) does not hold, then the key $K$ does not come from NODE $i$; else, NODE $i$ and NODE $j$ share the key $K$.

**END.**

Consider the following unrealistic example for illustration. Take $q = 23$ and $\alpha = 5$ (by computing the modular powers of 5, it can be seen that they generate every non-zero element in $Z(23)$, and therefore $i$ is a primitive element of this field). NODE $i$ and NODE $j$ can secretly pick for example $ti = 11$ and $tj = 18$, so that $\text{ID}_i = 5^{11} \mod 23 = 22$ and $\text{ID}_j = 5^{18} \mod 23 = 6$. Then in the first step of the algorithm, NODE $j$ chooses for instance $X_1 = 15$ (15 and $q - 1 = 22$ are relatively prime) and it has $Y_1 = 5^{15} \mod 23 = 19$; $Y_1$ and $\text{ID}_j$ are sent to NODE $i$. In the second step, NODE $i$ chooses $X_2 = 13$ and $X_4 = 20$ so that eqn. (2) holds: $X_2 + X_4 \mod 22 = 13 + 20 \mod 22 = 11 = ti$; then this node computes $K = 5^{13} \mod 23 = 21$, $Y_2 = 19^{13} \mod 23 = 7$ and $Y_4 = 19^{20} \mod 23 = 13$ and sends the last two values to NODE $j$. Finally, in the third step, NODE $j$ uses for example Euclid's algorithm to find $X_1^{-1} = 3$ such that eqn. (3) holds (i.e. $3 \cdot 15 \mod 22 = 1$) and computes $K = Y_2^3 \mod 23 = 7^3 \mod 23 = 21$, which is the same value for $K$ computed by NODE $i$ in the second step; the authentication test also holds, because $Y_4^3 \mod 23 = 13 \mod 23 = 12$ and $K \cdot 12 \mod 23 = 21 \cdot 12 \mod 23 = 22 = \text{ID}_i$.

In order to evaluate the interactive complexity of the algorithm, it should be noted that some computations can be accelerated both at NODE $i$ and at NODE $j$ by making use of precalculation:

(A) NODE $i$ can pregenerate some $(X_2, X_4)$ pairs of the next key transmissions to other nodes, and can precalculate the keys $K$ it will send. In this way, step 2 is accelerated.
(B) While waiting for Y2, NODE j can compute X1 - 1, so that step 3 is faster. Alternatively, a precalculation of some (X1, X1 - 1, Y1) triples is more efficient.

With the improvements (A) and (B) just four interactive exponentiations are required to complete the key exchange:

(1) To compute Y2 at NODE i.
(2) To compute Y4 at NODE i.
(3) To retrieve K at NODE j.
(4) To retrieve aX4 mod q at NODE j.

If both nodes are multiprocessor, (1) and (2) can be parallelized at NODE i, as well as (3) and (4) at NODE j. In this way, only the time for two interactive exponentiations is needed.

5. A Signature Scheme

When a number K in the form K = a^c mod q is available to both NODE i and NODE j (with c being only known to NODE i, for example after using the key exchange algorithm described in the previous section), it is possible to derive a signature scheme. It is worthwhile noting that K does not need to be secret, but just authentic, since it plays the role of a public key. Some public agreements must be added to those discussed in Section 3.

A small integer r(< q - 1) and a mapping function f

f: R → S

are publicly known. R and S are also public sets: R contains r integers such that no two among them yield the same power of K and S is a set of r bit sequences (for example, r = 4, R = {3, 5, 7, 11} and S = {"00", "01", "10", "11"} as proposed above (r = 4 is a realistic value since a small r is required in general); suppose f(3) = "00", f(5) = "01", f(7) = "10", f(11) = "11". Modular exponentiation of K to the exponents in R yields no two equal powers: 15, 14, 10, 22. Now suppose NODE i wants to sign the bit sequence (or message) "10" represented by m = 7 (among the four possible messages). In the first step, NODE i picks X2 = 17

**STEP 1. NODE i:** Pick X2 and X4 such that

\[ X2 + X4 = c \mod (q - 1) \]

\[ 2 < = X2, X4 < = q - 2 \]

(7)

where c is the power of a yielding K.

Compute \( Y2' = a^{mX2} \mod q \)

Send \( Y2' \) and \( X4' = mX4 \mod (q - 1) \) to NODE j.

**STEP 2. NODE j:** Compute

\[ Y2' \cdot a^{X4'} \mod q = a^{m(X2 + X4)} \mod q \]

\[ = a^{m} \mod q \]

\[ = K^{m} \mod q \]

(8)

Assume now that NODE j has a precalculated table with the powers of K corresponding to the exponents in R, so that it can recover the f(m) bit sequence NODE i sent.

**END.**

In order to thwart linear attacks on the signature scheme (see Section 6.2), choose the numbers in R to be relatively prime and change (X2, X4) for each message.

Let us also illustrate the operation of this algorithm with some trivially small figures. Take again q = 23, a = 5 and K = 21 so that c = 13 (this value, such that 5^13 mod 23 = 21, is assumed to be known only to NODE i). Choose r = 4, R = {3, 5, 7, 11} and S = {"00", "01", "10", "11"} as proposed above (r = 4 is a realistic value since a small r is required in general); suppose f(3) = "00", f(5) = "01", f(7) = "10", f(11) = "11". Modular exponentiation of K to the exponents in R yields no two equal powers: 15, 14, 10, 22. Now suppose NODE i wants to sign the bit sequence (or message) "10" represented by m = 7 (among the four possible messages). In the first step, NODE i picks X2 = 17
and $X_4 = 18$, so that eqn. (7) holds (i.e. $17 + 18 \mod 22 = 13 = r$); then it computes $Y_2' = 5^{17 \times 18} \mod 23 = 11$ and $X_4' = 7 \times 18 \mod 22 = 16$, which constitute the signed form of $m$ sent to NODE $j$. In the second step NODE $j$ calculates $Y_2'' = 5^{X_2''} \mod 23 = 11$ and $X_4'' = 7 \times 18 \mod 22 = 16$, which constitute the signed form of $m$ sent to NODE $j$. In the second step NODE $j$ calculates $Y_2'' = 5^{X_2''} \mod 23 = 11$, and NODE $j$ is supposed to store the four powers of $K$ to the exponents in $R$, $15$, $14$, $10$, $22$, and hence may check that the signature is a valid one, corresponding to the message represented by $m = 7$, that is “10”. It should be noted that because of $q$ having been chosen to be so small, the probability of a random successful hit is $r/q = 4/23$; if a non-trivial $q$ (such that $r \leq q$) is used, this probability is no longer meaningful.

6. Attacking the Key Exchange Protocol and the Signature Scheme

6.1 Attacks to the Key Exchange
Screcy is based on the discrete logarithm problem. Without knowledge of $X_1$, it is very hard to retrieve $K$ from $Y_2$ and $Y_4$. And finding $X_1$ from $Y_1$ presents any spy with the logarithm problem. Thus $K$ will remain only known to NODE $i$ and NODE $j$.

Authentication is guaranteed by the difficulty to produce a $Y_4$ that passes the authentication test at NODE $j$. It is nonsense for NODE $j$ itself to try and forge a key. On the other hand, consider the following attack.

(1) An eavesdropper $E$ allows STEP 1 to complete normally.

(2) $E$ intercepts NODE $i$'s response ($Y_2$, $Y_4$) in STEP 2, picks a $X_2''$ at random and computes

$$Y_2'' = Y_1^{X_2''} \mod q$$

$$K'' = a^{X_2''} \mod q$$

$$Y_4'' = Y_2 \times Y_4 / Y_2''$$

Then $E$ substitutes $Y_2''$ for $Y_2$ and $Y_4''$ for $Y_4$, and sends them to NODE $j$.

(3) $Y_2''$ and $Y_4''$ pass the authentication test of NODE $j$ in STEP 3.

Some provisions must be taken in order to discourage this kind of attack (it should be noted that $E$ cannot recover the good $K$):

First, the key $K$ must be unidirectional (to be used only from NODE $i$ to NODE $j$). It should be noted that most classical key management schemes use unidirectional keys (see ref. [11], for example).

Secondly, only one key exchange between NODE $i$ and NODE $j$ will be allowed (NODE $i$ adds the local label $Y_1$ to ID$^j$ which means that NODE $j$ has already been given a key); if another node has obtained a key by impersonating NODE $j$, this node can complain using a signed message. To change the current key, encipher the new key (mixed with some filler plaintext if necessary) under the old key by using a conventional cryptosystem.

Because of the first requirement, $E$ can get at most a pair of unidirectional keys ($K''_E$, $K''_j$), so as to be able to send forgeries to NODE $i$ and to NODE $j$. However, $E$ is unable to understand the answers of NODE $i$ and NODE $j$ ($E$ cannot recover the good keys), which makes it easy for these nodes to detect $E$.

The second requirement ensures that it will not be possible for $E$ to obtain a key by impersonating NODE $j$. NODE $i$ will not deliver a key after receiving a second request labeled ID$^j$. If this second request comes from NODE $j$ itself, this node can immediately complain by means of a signed message, thus causing NODE $i$ to invalidate

---

1 We are implicitly assuming that all keys in the network are exchanged previous to normal message transmission. If this is not true, the impersonator of NODE $j$ can also be detected by starting a message handshake (because of the two provisions taken, the impersonator cannot obtain a NODE $j$'s emitting key). Alternatively, only the first provision is needed if we take always $Y_1 = ID^j$ and $X_1 = j$; then we lose some randomization, but impersonation is never possible.
the first key sent to the eavesdropper E. On the other hand, if E sends a request labeled IDj to NODE i after the true NODE j has been given a key, nothing will happen, since E is not able to produce a signed complaint.

6.2 Attacks to the Signature Scheme
Here, authentication relies on the ability to produce a $X4$ such that the authentication test (8) at NODE j holds (remember that c is only known to NODE j). In a random attack, r values out of $q$ for $K^a$ will pass (8); since $r < q$, the probability of a successful forgery is really low. $X4$ is sent instead of a power of $a$ (as it was done in the key exchange protocol), so as to thwart the attack discussed in Section 6.1; this attack can be harmful when dealing with data instead of exchanging unidirectional keys.

Given signatures for messages $m_1$ and $m_2$, it is not possible to sign $m_1 + m_2$ if the $(X2, X4)$ pair is changed for each message. An attempt at signing $k \cdot m_1$ for some integer $k$ will equally fail if the numbers in $R$ are relatively prime ($k \cdot m_1$ will be an illegal message). Hence, it is not possible to sign a linear combination of known signed messages.

7. A Comparison with Some Other Methods
For the key exchange we will consider the following.

Diffie-Hellman exponential key exchange ([4]). When A and B want to exchange a key using this method, A chooses a random number $Xa$ in the range $2 \cdots q - 1$, where $q$ is a large prime. A keeps $Xa$ secret and sends $Ya = a^{Xa}$ mod $q$, with $a$ a fixed primitive element of $Z/q$ (a, $q$ are publicly known). Similarly, B picks an $Xb$ and sends $Yb$ to A. Then A computes $Kab = Yb^{Xa}$ mod $q$; B computes the same $Kab$ by letting $Kab = Ya^{Xb}$ mod $q$. Numerically, if $q = 23$ and $a = 5$, then it is possible for A and B to secretly choose $Xa = 10$ and $Xb = 13$ respectively; then A computes and sends $Ya = 5^{10}$ mod $23 = 9$, and B does the same with $Yb = 5^{13}$ mod $23 = 21$. Now, A retrieves the key $Kab$ by letting $Kab = Yb^{Xa}$ mod $23 = 21^{10}$ mod $23 = 12$; the same key is retrieved by B since $Ya^{Xb}$ mod $23 = 9^{13}$ mod $23 = 12 = Kab$.

No one but A and B can compute the key $Kab$, but an impersonation attack is feasible: a spy C can fool A and B, thus getting a $Kac$ and a $Kbc$, so that it will be possible for C to listen and to speak. It should be noted that unidirectional keys cannot be used here, since the Diffie-Hellman system is a symmetric one.

On the other hand, if $Ya$ and $Yb$ are fixed in a public file in order to avoid on-line attacks, then $Xa$ and $Xb$ are also fixed (and only known to A and B respectively). Hence, the key to be exchanged between NODE i and NODE j is unique and has a constant value $Kab$.

Our method avoids the impersonation attack, and allows NODE i to set an arbitrary key (this is useful for unidirectional keys).

RSA ([12]). Four exponentiations are necessary for an authentic and secret key exchange (encrypt-sign-decrypt-verify signature), like with our method. However, these four exponentiations cannot be parallelized, whereas in a multiprocessor system this is possible with our algorithm. Besides, RSA requires a lengthy strong prime computation.

For the signature scheme (not the key exchange), remember the ElGamal signature:

ElGamal ([7]). An ElGamal signed message is three times longer than the original message (the signature is twice as long as the message being signed). In our case, a signed message includes the message itself in an encoded form, so that a message signed with our system is just twice as long as the original message. Furthermore, signing and verifying a signature with ElGamal involves four exponentiations, one for signing and three for verifying the
signature, while only two exponentiations are required by our signature scheme.

8. Conclusions

As already outlined, the key exchange algorithm provides keys which can be easily converted to DES-like keys. Alternatively, after a key $K$ has been exchanged in an authentic way, we can use it for the algorithm of Section 5 to work as a signature scheme ($K$ can be public from now on). It should be noted that the authentication mechanism behaves in a similar manner to a zero-knowledge proof [8, 9], but as far as it is known to the authors, the practical issue is a new one, or at least an original one.

The small interactive computation, the similarity of key exchange and signatures, and the flexibility provided by the function $f$ (eqn. (6)) are some appealing properties of the proposed scheme. Perhaps simplicity and speed are achieved at the expense of a small increase in the required storage, which in any case is not really meaningful. On the other hand, breaking this system seems to be equivalent to solving the discrete logarithm problem, but as in ref. [7] no formal proof of this is given in this paper. However, when justifying the secrecy and authentication features, no other way to break the algorithm is evident.

All the described possibilities enable us to start up a secure network efficiently without using any previously shared secret information. In our opinion, this fact makes the proposed scheme very attractive when dealing with time critical and insecure channels, since it reduces computation and eliminates the need for trustees external to the system.

References

Josep Domingo-Ferrer received the diploma and the M.Sc. degrees in computer science from the Universitat Autònoma de Barcelona (Autonomous University of Barcelona) in 1988 and in 1989 respectively, both with honors. He also earned a B.Sc. degree in mathematics from the UNED, Madrid, 1989. His Master's thesis was entitled "SICNET: a model for a Securely Installable Cryptographic Network" and presented a prototype secure communications network, combining some original algorithms with many of the latest techniques in communications security.

Mr. Domingo-Ferrer has authored several papers on computer security and cryptography (MIMI '88, Eurocrypt '89, CO89). His research interests include secure communications as well as the connection between cryptography and classical data security (access control, inference control, flow control). He is currently working for a Ph.D. degree in computer security, with a postgraduate research scholarship from the Spanish Government. (Departament d'Informàtica, Universitat Autònoma de Barcelona, 08193 Bellaterra, Barcelona, Catalonia-Spain).

Llorenç Huguet-Rotger received the diploma degrees in mathematics and in applied sciences from the Universitat Autònoma de Barcelona and from the Université Catholique de Louvain (Belgium), in 1977 and in 1980 respectively. In 1981, he received a Ph.D. degree in computer science from the Universitat Autònoma de Barcelona.

Dr. Huguet-Rotger joined the Computer Science Department of the Universitat Autònoma de Barcelona in 1977. Since then, his main research areas have been in error-correcting codes and cryptography. He has delivered a number of papers on these topics and has been in the program committees of several related conferences. He has published a textbook in coding theory entitled Codes Correcteurs: Théorie et Applications, Masson, 1988; he also was a co-editor of Applied Algebra, Algebraic Algorithms and Error-Correcting Codes (Proceedings of the 5th AAECC International Conference), Springer-Verlag Lecture Notes in Computer Science, 1989. (Departament d'Informàtica, Universitat Autònoma de Barcelona, 08193 Bellaterra, Barcelona, Catalonia-Spain).