Cryptanalysis of an Authentication Protocol

Bogdan Groza
"Politehnica" University of Timisoara
Automation and Applied Informatics Department
Bd. Vasile Parvan nr. 2,Timisoara, Romania
E-mail: bogdan.groza@aut.upt.ro

Dorina Petrica
"Politehnica" University of Timisoara
Automation and Applied Informatics Department
Bd. Vasile Parvan nr. 2,Timisoara, Romania
E-mail: dorina.petrica@aut.upt.ro

Abstract

Authentication protocols have applications in many fields. The security of authentication protocols is commonly based on cryptographic primitives. Constructing secure authentication protocols is not an easy challenge and there is a large number of authentication protocols that prove to be insecure. Wulf et al. have proposed in [1] a protocol by which an entity can authenticate in a distributed system environment without using any shared secret. This paper will make a brief analysis of the proposed protocol and will show how it can be broken. The protocol will be broken by using multiplicative inverses of the integers from Z_n, where n is a prime number.

1. Introduction

In this paper we will analyze the security of the authentication protocol proposed by Wulf et al. in [1]. Some of the ideas from [1] were also proposed more recently in the case of a publish-subscribe system [2].

Authentication protocols are probably the most commonly used security protocols. Entity authentication is a process in which an entity proves his identity and his presence to another entity. Authentication requires both an identity guarantee, which is usually connected to the presence of a secret (for example a password), and a time guarantee which will be made by some time variant parameters - to ensure that this authentication did not happened before. Authentications are usually challenge-response protocols in which an entity sends a random challenge to another entity who wishes to prove his identity.

The security of authentication protocols was much debated especially in the last decade when many protocols that were known to be secure failed under a more careful analysis.

There is long list of protocols that were proved to be insecure; probably the most well known cases are that of Needham-Schroeder, Otway-Rees, TMN. Finally, the most important thing is that understanding why security fails has an important role in constructing stronger solutions.

In this paper we will show that the proposal from [1] can be broken easily only by using multiplicative inverses of integers from Z_n, where n is a prime number.

Section 2 shows the description of the protocol from [1] and section 3 outlines some mistakes in the proposal. In section 4 the protocol is analyzed while section 5 shows how to break the protocol. In section 6 some additional remarks will be made and section 7 holds the conclusion of the paper.

2. The Description of the Protocol

In order to build an authentication protocol, the use of two functions f, g with the following properties was proposed in [1]:

$$f \text{ and } g \text{ are one-way functions (1)}$$

$$g(f(x), f(y)) = f(g(x, y)) \quad (2)$$

The fact that these functions are one-way means that by giving x it is easy to compute $f(x)$ but by giving $f(x)$ it is infeasible or hard to compute x.

Functions f and respectively g are chosen in secret by each entity and will remain secret on the respective side. One entity will use g to compute a challenge for the other while the response can be computed only by the other entity which is in

Proceedings of the Seventh International Symposium on Symbolic and Numeric Algorithms for Scientific Computing (SYNASC'05)
0-7695-2453-2/05 $20.00 © 2005 IEEE

possession of f. The notations A and B will be used to denote the entities and it will be considered that A needs to authenticate to B. In the first step A will send to B a value x and the value of $f(x)$. This will be the first step:

Step1. A→B: A, x, $f(x)$

Now, in order to authenticate to B, A will send a request and B will send him back a challenge which consist on a randomly selected number y and the value of $g(x,y)$. In order to respond to this challenge A will compute $f(y)$ and $f(g(x,y))$. The following are the steps of the authentication protocol:

Step 2. A→B: A
Step 3. B→A: y, $g(x,y)$
Step 4. A→B: $f(y)$, $f(g(x,y))$

Now B has to verify that A is authentic and he achieves this by computing $g(f(x),f(y))$ and checking that indeed $g(f(x),f(y)) = f(g(x,y))$.

The reader may now easily remark why it was necessary for f and g to remain secret: in case that f is known the attacker can easily respond to the challenge by computing $f(y)$, $f(g(x,y))$; otherwise if g is known to the attacker he can respond with $r, g(f(x),r)$ for some arbitrary value r and the authentication will also hold.

In order to increase the effectiveness of the protocol it was also proposed that verifier values which is the pair $x, f(x)$ to be replaced in every protocol run with the newly sent values $y, f(y)$ or else with $g(x,y), f(g(x,y))$. In section 4 we will show how this replacement will lead an attacker to both impersonate A and make impossible for A to authenticate ever again.

This was the formal description of the protocol and in order to obtain a practical implementation the use of two functions based on discrete exponentiation was proposed. The functions proposed in [1] were:

$$f(x) = x^a \bmod n \quad (3)$$

$$g(x,y) = (x \cdot y)^b \bmod n \quad (4)$$

These functions will remain secret in the sense that each entity will choose an arbitrary value for a and respectively for b and keep the value secret.

These functions are suitable to be used in the proposed authentication scheme. Both functions will be hard to be deduced by an adversary because in order to determine a or respectively b the adversary had to compute a discrete logarithm. Therefore, the secrecy of these functions relies on the difficulty of computing discrete logarithms in Z_n. It is commonly known that the discrete logarithm is infeasible to compute, this means that giving $x^b \bmod n$, x, n it is infeasible to compute b for large values of n.

The composition property (2) is also easy to verify since indeed $g(f(x),f(y)) = \left(x^a \cdot y^a\right)^b \bmod n =$ $= f\left(g\left(x,y\right)\right)$.

The following additional conditions were imposed for the parameters: n, a and b are all large primes.

With the previously defined functions f and g the protocol can be rewritten as follows:

Step 1. A→B: A, x, $f(x) = x^a \bmod n$
Step 2. A→B: A
Step 3. B→A: y, $g(x,y) = (x \cdot y)^b \bmod n$
Step 4. A→B: $f(y) = y^a \bmod n$, $f\left(g\left(x,y\right)\right) =$
$= \left((x \cdot y)^b\right)^a \bmod n$

The identity of A will be verified by computing $g(f(x),f(y)) = \left(x^a \cdot y^a\right)^b \bmod n$ and checking that $g(f(x),f(y)) = f(g(x,y))$.

3. Some Shortcomings of the Proposal

We will now outline some shortcomings of the proposal from [1]:

1) The conditions imposed for functions f and g are not sufficient to guarantee security. One may note that the protocol can be broken if an attacker can determine for a given value γ two values α and β such that $g(\gamma,\alpha) = \beta$. This attack is possible because in the verification step B will compute and verify the value of $g(\gamma,\alpha) = \beta$ for the newly received values $\alpha = f(y)$, $\beta = f(g(x,y))$ and the verifier $\gamma = f(x)$ - so if such α and β can be determined the authentication holds. This additional condition should

be stated in [1] and also in [2] because there is no proof that the previous two conditions (1),(2) for functions f and g can guarantee the security of the protocol. The attack in section 4 is based on this remark.

2) The condition imposed for the values of n, a and b to be large primes might not be appropriate for achieving stronger security. It is natural to choose n a large prime because it is needed to make the discrete logarithm intractable. But we do not see any argument for requesting a and b to be also primes. More, we see that in fact it should be requested for x to be a generator of Z_n in order to make the discrete logarithm hard to compute – otherwise if the order of x is small it will be much easier to compute the discrete logarithm. If the verifier is always changed it is also recommended for the value of the verifier to have a high order because again the intractability of the discrete logarithm depends on this order.

3) It is possible for an attacker to mount chosen-text attacks against the protocol; these are attacks in which an adversary strategically chooses challenges in order to learn information about the secret. The attacker can lunch such an attack by selecting some elements from Z_n with low order. The attacker can take some divisors $\delta_i, i=\overline{1,k}$ of the order group of the group Z_n (i.e. $\delta_i \mid n-1,\ i=\overline{1,k}$ when n is prime) such that they are relatively prime to each other (i.e. $\gcd(\delta_i,\delta_j)=1,\ i\neq j$, gcd denotes the greatest common divisor) and it is feasible to compute discrete logarithms on elements with order $(n-1)/\delta_i, i=\overline{1,k}$. Let χ be a generator of Z_n and compute elements $\omega_i=\chi^{\delta_i} \bmod n, i=\overline{1,k}$, obviously each ω_i will have the order $\zeta_i=(n-1)/\delta_i, i=\overline{1,k}$. Suppose that the values of ω_i are offered as challenges to entity A. Since it will be feasible to compute discrete logarithms on each element ω_i the adversary can learn from the responses of A the congruence of the secret exponent modulo each ζ_i. This leads to congruencies of the form $a\equiv\lambda_i \bmod \zeta_i,\ i=\overline{1,k}$ which results in a system that can be solved by Chinese Remaindering Theorem and has a unique solution modulo $\prod_{i=1}^{k}\zeta_i$. If we compute the values of $\sigma_i=(\rho_i)^{-1} \bmod \zeta_i$ and $\rho_i=\frac{\prod_{i=1}^{k}\zeta_i}{\zeta_i}$ then the solution of these congruencies will be $\tau=\sum_{i=1}^{k}\lambda_i\cdot\rho_i\cdot\sigma_i$ and for the secret exponent holds $a\equiv\tau \bmod \prod_{i=1}^{k}\zeta_i$.

4) The authors sustained that there is a connection between this protocol and zero knowledge protocols. Zero-knowledge protocols are build to address the fact that an adversarial verifier may be able to select challenges in such a way that he can obtain responds that provide certain information about the secret on which authentication is made (i.e. chosen text attacks). In this protocol an adversarial verifier can strategically give such challenges, as seen previously, so this protocol is certainly not zero-knowledge. In the form presented and with the selected candidates for functions f and g this is certainly a public key challenge-response authentication (section 6 examines the similarities between this protocol and a public key protocol).

4. Analyzing the Protocol

A careful analysis of the proposed functions will be done.

First we will remark an additional property of g that was not taken into account in [1], observe that:

$$g(x,y)=g(x\cdot y,1) \quad (5)$$

Although it may appear trivial this property has some fundamental implications.

The most important is that the claim that an adversary cannot compute $g(x,y)$ for some arbitrary values of x and y is not always true. It is easy to observe that if one knows $g(a,b)$ he also knows $g(a\cdot b\cdot\xi,\xi^{-1})$ for any integer ξ, here ξ^{-1} denotes the multiplicative inverse of ξ. Now, because of (5) the following relation holds:

$$g(a\cdot b\cdot\xi,\xi^{-1})=g(a\cdot b,\xi\cdot\xi^{-1})=g(a,b) \quad (6)$$

Define now the function ψ_k, where k is a constant integer:

$$\psi_k(x) = x^k \bmod n \quad (7)$$

Observe also that ψ has the following property:

$$\psi_k(x \cdot y) = \psi_k(x) \cdot \psi_k(y) \quad (8)$$

Redefine f using ψ:

$$f(x) = \psi_a(x) \quad (9)$$

Also remark that function g is from the same family as f and the fact that it has two parameters it is not so important since:

$$g(x, y) = \psi_b(x \cdot y) \quad (10)$$

Now, by using ψ redefine the authentication steps as follows:

Step1. A→B: A, x, $\psi_a(x)$
Step 2. A→B: A
Step 3. B→A: y, $\psi_b(x \cdot y)$
Step 4. A→B: $\psi_a(y)$, $\psi_a(\psi_b(x \cdot y))$

And the verification will be done by checking that $\psi_b(\psi_a(x) \cdot \psi_a(y)) = \psi_a(\psi_b(x \cdot y))$. This is certainly a more elegant and realistic description of the protocol.

Define also:

$$[\psi_a(x)]^{-1} = \psi_a\left(x^{-1}\right) \quad (11)$$

By using (8) observe that:

$$[\psi_a(x)]^{-1} \cdot \psi_a(x) = \psi_a\left(x^{-1} \cdot x\right) = \psi_a(1) = 1 \quad (12)$$

It is now easy to see how an attacker can impersonate entity A, let ψ_c denote an arbitrary instance of ψ chosen by the attacker:

Step 2. Attacker→B: A
Step 3. B→Attacker: y, $\psi_c(x \cdot y)$
Step 4. Attacker→B: $\psi_a\left(x^{-1}\right) \cdot \psi_c(x \cdot y)$, $\psi_c(\psi_b(x \cdot y))$

The authentication of the attacker certainly holds since one may easily verify that indeed $\psi_b\left(\psi_a(x) \cdot \psi_a\left(x^{-1}\right) \cdot \psi_c(x \cdot y)\right) = \psi_c(\psi_b(x \cdot y))$.

If we remember the security issue from the previous chapter, we can see that this attack was possible because the adversary was capable to compute $\alpha = \psi_a\left(x^{-1}\right) \cdot \psi_c(x \cdot y)$ and $\beta = \psi_c(\psi_b(x \cdot y))$ such that for the given $\gamma = \psi_a(x)$ it holds that $g(\gamma, \alpha) = \beta$. Computation of such α and β was possible because property (6) let the adversary predict the result of g over the authentication values.

It worth also to note that the attack was possible because even if an attacker does not know the value of $\psi_a(x)$ he can make the verification not depend on it by computing the multiplicative inverse $\psi_a\left(x^{-1}\right)$ and since (12) holds the verification does not depend anymore on the value of the verifier $\psi_a(x)$.

As a brief conclusion the attacker can break the protocol if he has the ability to compute $\psi_a\left(x^{-1}\right)$ given $\psi_a(x)$ - this is certainly the case of this protocol and the next section will describe the concrete attack over the protocol.

5. How to Break the Protocol

By using the elements form the previous section it is easy to break the protocol.

We will remark that when n is prime Z_n forms a group with the operation of multiplication - this means that all elements have a multiplicative inverse. Multiplicative inverses are easy to compute in Z_n, whether n is prime or not, with the extended Euclidean algorithm [3, page 67].

Now an attacker can easily compute the values $\alpha = x^{-a} \cdot x^k \cdot y^k \bmod n$ and $\beta = \left((x \cdot y)^b\right)^k \bmod n$, here k is a random integer selected by the attacker. This is possible since x, x^a, y and $(x \cdot y)^b$ are not secret and the multiplicative inverse of x^a, that is x^{-a}, is easy to compute. The attacker will use these values α, β in the third step of the authentication.

The following would be the run of the protocol in the case when the attacker impersonates the real user (we will suppose that Step1 of the protocol was already run as in section 2):

Step 2. Attacker→B: A
Step 3. B→Attacker: y, $g(x, y) = (x \cdot y)^b \bmod n$
Step 4. Attacker→B: $\alpha = x^{-a} \cdot x^k \cdot y^k \bmod n$,
$\beta = \left((x \cdot y)^b\right)^k \bmod n$

The attacker has successfully impersonated A and the verification of the identity holds because it is easy to check that indeed $g(\alpha, x^a) = (x^{-a} \cdot x^k \cdot y^k \cdot x^a)^b \bmod n = \beta$. Again notice that even if an attacker does not know the value of a he can make the verification not depend on it by computing the multiplicative inverse of x^a and since $x^a \cdot x^{-a} = 1$ - the authentication protocol does not depend anymore on the verifier x^a.

Example: In order to make things more clear we will consider this example. Suppose that A chooses the prime $n = 12457$, $a = 2351$, $x = 509$ while B will choose $y = 673$, $b = 2953$. The first step will be:

Step1. A→B: A, $x = 509$, $x^a \bmod n = 778$

Now the attacker request an authentication and B sends him a challenge:

Step 2. Attacker→B: A

Step 3. B→Attacker: $y = 673$, $(x \cdot y)^b \bmod n = 1074$

The attacker chooses an arbitrary $k = 253$, computes the multiplicative inverse of $x^a \bmod n$ (that is $x^{-a} \bmod n = 9687$ since $x^{-a} \cdot x^a = 778 \cdot 9687 \equiv 1 \bmod n$) and responds with the following values:

Step 4. Attacker→B: $\alpha = x^{-a} \cdot x^k \cdot y^k \bmod n = 7586$, $\beta = \left((x \cdot y)^b\right)^k \bmod n = 6795$

Entity B verifies authenticity by checking that $\left(\alpha \cdot x^a\right)^b \bmod n = \beta$ and this holds since $(7586 \cdot 778)^{2953} \bmod 12457 = 6795$.

Now if the values of the verifier are always replaced with the newly sent values it is easy to see that after an attack the real user will never be able to authenticate again. For example if we suppose that the previous attack took place and B had replace the verifier pair x, x^a with the newly sent values $y, \alpha = x^{-a} \cdot x^k \cdot y^k \bmod n$ (about which B believed although cannot verify that is $y^a \bmod n$) when A request an authentication and B sends him a new challenge y' the following will be steps of the authentication:

Step 5. A→B: A

Step 6. B→A: y', $g(y, y') = (y \cdot y')^b \bmod n$

Step 7. A→B: $f(y')$, $f(g(y, y')) = \left((y \cdot y')^b\right)^a \bmod n$

The authentication fails and A will be rejected because B will verify that $g(\alpha, f(y')) = f(g(y, y'))$ and B will decide that A is not authentic since $\left(x^{-a} \cdot x^k \cdot y'^a\right)^b \neq \left((y \cdot y')^b\right)^a \bmod n$.

Example: Consider the previous attack on the protocol, now the verifier is $y = 673$, $\alpha = 7586$. Entity A wants to authenticate and B sends the challenge $y' = 129$, the protocol run will be the following:

Step 5. A→B: A

Step 6. B→A: $y' = 129$, $g(y, y') = 11708$

Step 7. A→B: $f(y') = 9969$, $f(g(y, y')) = 3272$

Now B will verify that A is authentic by checking that $(\alpha \cdot f(y'))^b = f(g(y, y'))$ but since $(\alpha \cdot f(y'))^b = 4444$ the authentication will certainly fail.

Also if the verifier pair x, x^a is replaced by $g(x, y) = (x \cdot y)^b \bmod n$, $\beta = \left((x \cdot y)^b\right)^k \bmod n$ the authentication of A will again fail. The following will be the steps of the authentication in this case:

Step 5. A→B: A

Step 6. B→A: y', $g(g(x, y), y') = \left((x \cdot y)^b \cdot y'\right)^b \bmod n$

Step 7. A→B: $f(y') = (y')^b \bmod n$, $f(g(g(x \cdot y), y')) = \left(\left((x \cdot y)^b \cdot y'\right)^b\right)^a \bmod n$

The authentication fails because B will verify that $g(\beta, f(y')) = f(g((g(x \cdot y)), y'))$ and this is false since: $\left(\left((x \cdot y)^b\right)^k \cdot y'^a\right)^b \neq \left(\left((x \cdot y)^b \cdot y'\right)^b\right)^a \bmod n$.

Example: We will again consider the first attack on the protocol, now the verifier is $g(x, y) = 1074$ and $\beta = 6795$. Entity A wants to authenticate and B sends

the challenge $y' = 129$, the protocol run will be the following:

Step 5. A→B: A
Step 6. B→A: $y' = 129$, $g\left(g(x, y), y'\right) = 4364$
Step 7. A→B: $f\left(y'\right) = 9969$, $f\left(g\left(g(x \cdot y), y'\right)\right) = 4206$

In order to check that A is authentic B will verify that that $\left(\beta \cdot f\left(y'\right)\right)^b = f\left(g\left((g(x \cdot y)), y'\right)\right)$ but since $\left(\beta \cdot f\left(y'\right)\right)^b = 5027$ the authentication will certainly fail.

6. Some Additional Remarks

It is commonly known that any public-key encryption technique may be used to construct an authentication protocol and there are mainly two ways to achieve this [1, page 403]:

1) by making the claimant to decrypt a challenge encrypted with his public key
2) by making the claimant digitally sign a message with his private key

As an example, one can derive the following authentication protocol from ElGamal public key encryption [4] (consider the same function ψ from the previous section and that Step 1 has run in the same manner proposed in section 3):

Step 2. A→B: A
Step 3. B→A: $\psi_b(x)$, $y \cdot \psi_b(\psi_a(x))$
Step 4. A→B: y

In Step 3 A will decrypt the challenge y by computing $y = y \cdot \psi_b(\psi_a(x)) \cdot [\psi_a(\psi_b(x))]^{-1}$ and he sends back the value of y proving that he can decrypt arbitrary messages encrypted under its public key.

It should be also stated that such constructions should be carefully done since the use of the same private-public key pair for different purposes can result in security loss.

The proposal from [1] might appear interesting but with the candidates proposed it is nothing more than a challenge-response authentication based on a public-key primitive. It is obviously that the authentication from [1] is very similar to the previous authentication based on El-Gamal public key encryption, in fact the authentication from [1] requires much more computation and it is insecure.

Once defined the function ψ (see section 4) it is also easy to observe that:

$$\psi_b(\psi_a(x)) = \psi_a(\psi_b(x)) \quad (13)$$

We will now rewrite the steps of the authentication from section 3 in order to make some observations (remember that this is the authentication from [1] redefined by using ψ, Step1 has run as in section 2):

Step 2. A→B: A
Step 3. B→A: y, $\psi_b(x \cdot y)$
Step 4. A→B: $\psi_a(y)$, $\psi_a(\psi_b(x \cdot y))$

We will underline that the purpose of the challenge is to verify that A is capable to compute ψ_a for an arbitrary value. The authentication as described above, and with the selected candidates, contains redundant elements in Step 4 because A computes ψ_a for two distinct values and it will be sufficient to compute on only one value if there is a way to verify this. Notice that $\psi_b(x)$, if b is selected at random, is an arbitrary value and can be use in Step 2 as a challenge. Now the response of A to this challenge should be $\psi_a(\psi_b(x))$ and B could verify this since (13) holds and he can check that the response is equal to $\psi_b(\psi_a(x))$. So the protocol can be rewritten in the following simpler way:

Step 1. A→B: A, x, $\psi_a(x)$ - initialization
Step 2. A→B: A - request
Step 3. B→A: $\psi_b(x)$ - challenge
Step 4. A→B: $\psi_a(\psi_b(x))$ - response

But with the proposed functions this is nothing new but Diffie-Hellman key-exchange [4] in which A will prove that he can recover the key and therefore is authentic. So if redundant elements are removed from the authentication proposed in [1] it can be reduced to Diffie-Hellman key exchange.

7. Conclusions

The proposal from [1] has two main weak-points:

1) the description on a formal level of the two functions f and g with the requested properties cannot guarantee security
2) the selected functions were not carefully examined and some essential properties of integers from Z_n neglected

As a consequence the authentication protocol does not resist cryptanalysis and the attacker can successfully impersonate the user.

A new protocol can be added on the long and always open list of insecure protocols.

The proposal from [1] can appear interesting but with the selected candidates for functions f and g it results in an insecure authentication protocol. Finding some good candidates for these functions can finally lead to a functional authentication scheme – but this is the hardest task faced by cryptography.

8. References

[1] Wulf, W.A., Yasinsac, A., Oliver, K.S., Peri, R. "A technique for remote authentication." Availlable at http://www.cs.fsu.edu/~yasinsac/Papers/howdoi.pdf, last accessed 20.06.2005 20:00 PM.

[2] Wang, C., Carzaniga, A., Evans D., Wolf A.L., "Security issues and requirements for internet-scale publish-subscribe systems" Proceedings of the 35th Hawaii International Conference on System Sciences, (2002).

[3] Menezes, A.J., van Oorschot, P.C., Vanstone, S.A., "Handbook of Applied Cryptography" CRC Press, (1996).

[4] ElGamal, T., "A public key cryptosystem and a signature scheme based on discrete logarithms", IEEE Transactions on Information Theory, 31 (1985).

[5] Diffie, W., Hellman, M.E., "New directions in cryptography", IEEE Transactions on Information Theory, 22, (1976).