ONE TIME PASSWORDS FOR UNCERTAIN NUMBER OF AUTHENTICATIONS

Bogdan Groza, Dorina Petrica

"Politehnica" University of Timisoara Department of Automation and Applied Informatics
Bd. Vasile Parvan nr. 2, 302223 Timisoara, Romania
E-mail: bgroza@aut.utt.ro, dpetrica@aut.utt.ro

Abstract: Authentication based on cryptographic techniques is a subject of great interest in many fields. Authentication protocols presented in this paper are based on one-time passwords which offer stronger security than commonly used fixed passwords. Leslie Lamport in his paper Password Authentication with Insecure Communication proposed the use of one-way functions in order to obtain one time passwords. Because of their simplicity cryptographic hash functions are commonly used for such purpose. Some disadvantages of using hash-functions will be stated and then functions on groups of composite integers will be used in order to obtain a more flexible one-time password scheme.

Keywords: entity authentication, one-time password, uncertain number of authentications, bounded number of authentications, upper bound.

1. INTRODUCTION

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. In this paper we will use the term *user* to denote the entity which needs to authenticate and the term *system* to denote the entity to which identity is be proven.

Password authentication is the most commonly authentication. We use password to log on an operating system or to get money from our credit card etc. Using *conventional time-invariant passwords* has a major disadvantage: passwords can be stolen from the system where they are stored or by intercepting user's communication over insecure channels. A better solution is to use one-time passwords.

One-time passwords are passwords which are valid only once for an authentication. The main advantage in using them is that by disclosing an already used password the user may not be impersonated, since a one-time password may not be used twice.
Lamport (1981) has proposed a functional one-time password scheme in which secrets are stored only on the user side and intercepting a password sent from user to the system would not lead to an impersonation. Lamport's authentication is based on computing the sequence $\{x, F(x), F^1(x), F^2(x), ..., F^{N_A}(x)\}$ on the user side, where x is an arbitrary value chosen by the user and kept secret, N_A is the number of authentications to be performed and may be also chose by the user, F is a known one way function (this means that by giving x it is easy to compute $F(x)$ but by giving $F(x)$ it is infeasible to compute x).

At the beginning the system must know $F^{N_A}(x)$ and then when the user needs to authenticate for the first time to the system ($i=1$) he will present $F^{N_A-1}(x)$ as the first one-time password. At the i^{th} authentication the user will prove it's identity by sending $F^{N_A-i}(x)$ and the system will simply verify this by computing $F(F^{N_A-i}(x))$ and also checking that $F(F^{N_A-i}(x)) = F^{N_A-i+1}(x)$, where $F^{N_A-i+1}(x)$ is the previous authentic one time password. This scheme may also be viewed as a challenge-response protocol where the challenge is defined by the position of the password in the password sequence (Menezes et al., 1996, page 396). The authentication process is suggested in Figure 1.

Cryptographic hash functions are commonly used to ensure data integrity (Rivest, 1992a, 1992b; FIPS 180-1, 1995). Since hash-functions are one-way functions several papers propose the use of hash function in Lamport's scheme (Haller, 1995; Haller et. al., 1998). It should be also mentioned that block ciphers may also be used to generate a hash function if they behave like random functions (Menezes et al., 1996, page 328).

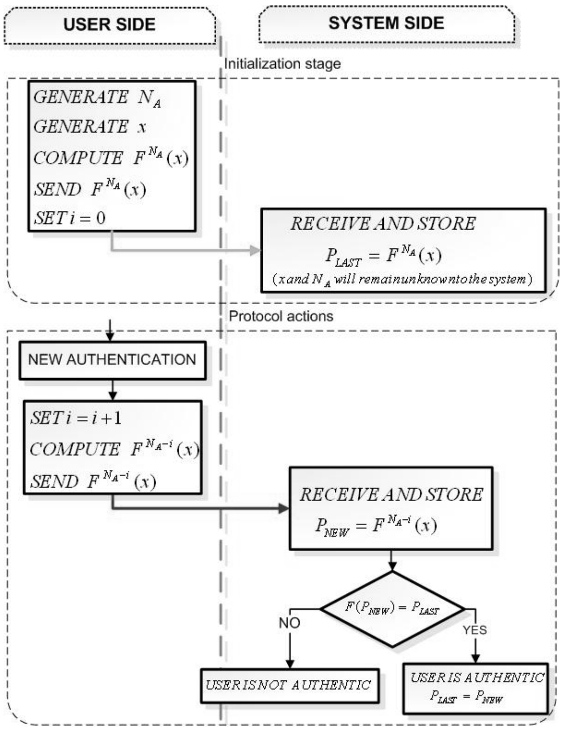

Fig. 1. Entity authentication with Lamport's one-time password scheme

It is obvious that the advantage of using one-time passwords is that they offer stronger security than fixed passwords. One-time password schemes are an important step to challenge-response entity authentication protocols. Nevertheless Lamport's scheme has some computational disadvantages. Other one-time password schemes were proposed, to remove those disadvantages, and some of them proved to be breakable (SAS, OSPA), others like ROSI are considered to be secure (Chien et al., 2003).

The most important disadvantage in Lamport's scheme with hash functions is the necessity to compute i composition of F in order to obtain $F^i(x), \forall i < N_A$ because this will require time to compute these compositions or space if the user decide to store all the passwords. This kind of disadvantage can be removed by using functions over Z_n^* which are more flexible, however these function have different disadvantages: they are harder to compute than simpler hash functions. The purpose of this paper is to propose some functions over Z_n^* which can generate one time passwords and to analyze some of their advantages and disadvantages.

In section 2 some disadvantages of Lamport's scheme with hash functions will be stated and in section 3 a more practical point of view will be set. Functions over Z_n^* will be used to generate one-time passwords in section 4 while section 5 will outline some advantages and disadvantages of these functions and section 6 will be the conclusion of this paper. Frequently used notations are presented in Appendix A.

2. SOME DISADVANTAGES IN LAMPORT'S SCHEME WITH HASH FUNCTIONS

The following issues may be viewed as disadvantages of using hash functions on Lamport's scheme:

- The number of authentications N_A allowed by the algorithm is a fixed number. If we need more authentications than we previously expected this will not be possible, because the algorithm does not allow, and a new array of one time passwords should be generated.
- The number of authentications N_A should be small enough to be reasonable to perform the computation of F^{N_A}. If we need N_A one-time passwords we will need to compute N_A compositions of F and this may be a time consuming operation and also may require space if the user decide to store the passwords. For example if $N_A = 2^{20}$ we will need to compute $F^{N_A} = F^{2^{20}}$ which will require more than one million compositions and time or space can became a problem if computational resources are limited.
- If the number of authentication is uncertain we may in fact not need N_A one-time passwords. For example we may consider that a user want to authenticate $N_A = 2^{20}$ times to a system during a

year and compute 2^{20} one-time passwords, but then if the user will authenticate only 10 times much of the computation was useless.

3. SOME PRACTICAL ASPECTS REGARDING THE NUMBER OF AUTHENTICATIONS

There are processes which may happen for an uncertain number of times. In real life there are many things which we could not know how many times will repeat. For example we may not know how many rainy days will be in an year, or how many times a user will log on to a computer in a month etc. In particular when we are referring to authentications we usually have to do with an *uncertain number of authentications*.

Even if we could not know how many times a process will happen this number is usually *bounded*. For example we might not know how many rainy days will be in a year but the number of rainy days is certainly bounded by 365 because there are at most 365 days in a year. Even if still uncertain we usually have to do with a *bounded number of authentications*. We will define the *upper bound* B_{U,N_A} of the number of authentications as an integer which is certainly bigger then the number of authentications N_A. We will not insist on how close is N_A to B_{U,N_A}, it is obvious that it is better to have B_{U,N_A} as close as possible to N_A but this is not of great relevance for this paper. We may also notice the following relation concerning B_{U,N_A}: $B_{U,N_A} < \frac{T}{T_A}$ where T – is the length of the time interval that we are referring to, and T_A is the time required for the authentication process to complete (we will suppose that two process cannot run at the same time, however it is easy to extend the paper to this case). The number of times an entity authenticates in a year is also bounded, for example if an entity needs to authenticate for one year, that is 365 days, and an authentication will require 30 seconds this means that $B_{U,N_A} < \frac{T}{T_A} = \frac{365 \cdot 24 \cdot 60 \cdot 60}{30} = 1.051.200 \cdot$

We will now remark that knowing the exact number of authentications N_A is usually not possible while knowing an upper bound B_{U,N_A} for this number is likely to be possible.

4. USING EXPONENTIATION OVER COMPOSITE MODULES

Functions defined over sets of integers, with inverses that are usually intractable, are frequently used for public-key authentications. We will use the composition of some of functions over Z_n^* to generate one-time passwords for a bounded number of authentications where bound B_{U,N_A} is very high.

The function $F(x) = x^{\varepsilon} \bmod n, \varepsilon \in Z$ defined over Z_n^* is the one-way trapdoor function used in the RSA public key encryption scheme. The function inverse could be computed only if the factorization of n is known and $\gcd(\varepsilon, \phi(n)) = 1$. We will therefore choose two random primes p and q, compute $n = p \cdot q$, a random exponent ε, a secret a and the one-time password array will be $A = \{a, F(a), F^2(a), F^3(a), ..., F^{\eta}(a)\}$ for a number of η authentications. The following remarks are relevant from some points of view:

1. From the security point of view:
1.1. Since the function inverse is intractable it should be impossible for an attacker who intercepts a one-time password to compute the next one-time password. This means that it should be infeasible to compute $F^i(x)$ from $F^j(x)$ with any $i < j$ for an adversary who does not know the factorization of n because the function inverse is intractable.
1.2. Functions over Z_n^* are cyclic and there exists i and j such that $F^i(x) = F^j(x), i \neq j$. This means that the sequence $\{a, F(a), F^2(a), F^3(a), ..., F^{\eta}(a)\}$ is cyclic for big values of η. However such cycles are unlikely to appear because it would lead to the factorization of the integer n and this integer will be chose large enough to make factorization infeasible (Menezes et al., 1996, page 289).

2. From the computational point of view:
2.1. It is not necessary to compute η compositions of $F(x)$ in order to obtain $F^{\eta}(x)$ because in Z_n^* exponents can be reduced modulo $\phi(n)$ and the following relation can be used:

$$F^{\eta}(x) = x^{\varepsilon^{\eta} \bmod \phi(n)} \bmod n \qquad (1)$$

Therefore only about $\frac{3}{2}\log_2\left(\varepsilon^{\eta} \bmod \phi(n)\right)$ multiplications over Z_n^* will be required to compute $F^{\eta}(x)$ (this is the expected running time of an exponentiation in Z_n^* (Menezes et al., 1996, page 614). Relation (1) is a consequence of the fact that $x^{\phi(n)} = 1 \bmod n, \forall x \in Z_n^*$ and since $F^{\eta}(x) = x^{\varepsilon^{\eta}} \bmod n$ then if we take $\varepsilon^{\eta} = \lambda + i \cdot \phi(n)$ it comes that $x^{\varepsilon^{\eta}} \bmod n = x^{\lambda + i \cdot \phi(n)} \bmod n = x^{\lambda} \cdot (x^i)^{\phi(n)} \bmod n = x^{\lambda} \bmod n \Leftrightarrow$ $x^{\varepsilon^{\eta}} \bmod n = x^{\varepsilon^{\eta} \bmod \phi(n)} \bmod n$.

2.2. As shown previously, knowledge of the factorization of n will be required in order to compute $F^{\eta}(x)$ as fast as possible, taking the advantage of $\phi(n)$ to reduce exponents. However generating the one-time password array could be done even without knowledge of the factorization of n, if the secret a is known, by using the composition of $F(x)$ and without taking advantage of the reduction of the exponents – this will result in a complete waste of time.

2.3. If $\gcd(\varepsilon, \phi(n)) = 1$ it is possible to compute the multiplicative inverse δ_{ε} of ε in $Z^{*}_{\phi(n)}$. This means that it is possible to compute a δ_{ε} such that $\varepsilon\delta_{\varepsilon} \equiv 1 + i \cdot \phi(n)$. Then $F^{\eta-1}(x) = (F^{\eta}(x))^{\delta_{\varepsilon}}$ because $(F^{\eta}(x))^{\delta_{\varepsilon}} = (x^{\varepsilon^{\eta}})^{\delta_{\varepsilon}} = x^{\varepsilon^{\eta} \cdot \delta_{\varepsilon}} = x^{\varepsilon^{\eta-1} \cdot \varepsilon \cdot \delta_{\varepsilon}} = x^{\varepsilon^{\eta-1}} \bmod n$. In other words the function inverse is $F^{-1}(x) = x^{\delta_{\varepsilon}} \bmod n$. This means that in this case computing the next password from the one just sent is possible if the factorization of n is known and $\gcd(\varepsilon, \phi(n)) = 1$.

In the general case when $F(x) = x^{\varepsilon} \bmod n, \varepsilon \in Z$ and the value of $\gcd(\varepsilon, \phi(n)) = 1$ is not used, the authentication process between a user and a system will run through the following stages also illustrated in Figure 2:

1. **Initialization stage:**

1.1. **User→System:** The user will generate: two random primes p and q, a random integer exponent ε, a random integer a, upper bound B_{U,N_A} (see section 2), then he computes: $n = p \cdot q, \phi(n) = (p-1) \cdot (q-1)$, $e = \varepsilon^{B_{U,N_A}} \bmod \phi(n)$, $F^{B_{U,N_A}}(a) = a^{e} \bmod n$. After setting $i = 1$ he will send secure to the system: $(n, F^{B_{U,N_A}}(a), \varepsilon)$

1.2. **System:** The system will receive and store: $(n, F^{B_{U,N_A}}(a), \varepsilon)$, $P_{LAST} = F^{B_{U,N_A}}(a)$

2. **Protocol actions.** For the authentication of the user to the system in the i^{th} session:

2.1. **User→System:** The user will set: $i = i + 1$ and compute the exponent and password: $e_i = \varepsilon^{B_{U,N_A} - i} \bmod \phi(n)$, $F^{B_{U,N_A} - i}(a) = a^{e_i} \bmod n$. Then he will send to the system: $F^{B_{U,N_A} - i}(a)$ as his new one-time password.

2.2. **System:** The system will receive and store: $P_{NEW} = F^{B_{U,N_A} - i}(a)$ and will check if $F(P_{NEW}) = P_{LAST}$. If this true then user is authentic and the system will set $P_{LAST} = P_{NEW}$ otherwise the user is not authentic.

It is easy to observe that indeed $F(P_{NEW}) = (a^{\varepsilon^{B_{U,N_A} - i}})^{\varepsilon} \bmod n = a^{\varepsilon^{B_{U,N_A} - i + 1}} \bmod n = P_{LAST}$.

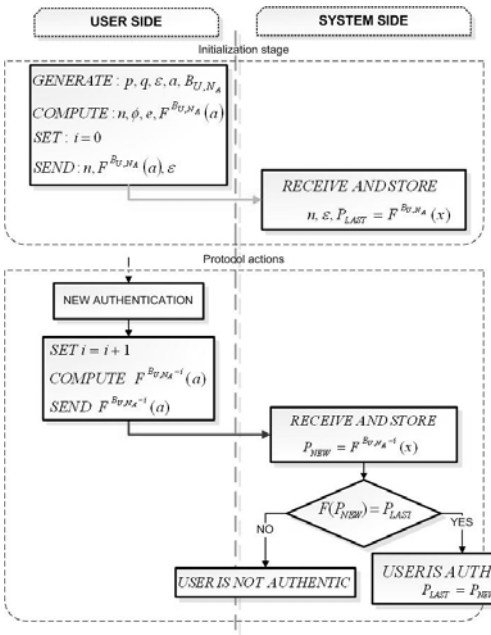

Fig. 2. One-time password authentication scheme with exponentiation over Z^{*}_{n}

As stated before if $\gcd(\varepsilon, \phi(n)) = 1$ there exists δ_{ε} such that $\varepsilon\delta_{\varepsilon} \equiv 1 \bmod \phi(n)$. In this particular case previous knowledge of the upper bound B_{U,N_A} is not needed and the authentication process between a user and a system will run through the following stages also illustrated in Figure 3:

1. **Initialization stage:**

1.1. **User→System:** The user generates: two random primes p and q, a random integer exponent ε such that $\gcd(\varepsilon, \phi(n)) = 1$, a random integer a, then he computes: $n = p \cdot q, \phi(n) = (p-1) \cdot (q-1)$, δ_{ε} such that $\varepsilon\delta_{\varepsilon} \equiv 1 \bmod \phi(n)$. After setting $i = 0$ he will send secure to the system: (n, a, ε)

1.2. **System:** Receive and store: (n, a, ε), $P_{LAST} = a$

2. **Protocol actions.** For the authentication of the user to the system in the i^{th} session

2.1. **User→System:** The user wil set: $i = i + 1$ then compute and send to the system his new one-time password: $a_i = a_{i-1}^{\delta_{\varepsilon}} \bmod n$ (where a_{i-1} is the previously sent password)

2.2. **System:** The system will receive and store: $P_{NEW} = a_i$ and will check if $F(P_{NEW}) = P_{LAST}$. If this is true then user is authentic and the system will set $P_{LAST} = P_{NEW}$, otherwise the user is not authentic.

It is easy to observe that indeed: $F(P_{NEW}) = a_i^{\varepsilon} \equiv \left(a_{i-1}^{\delta_{\varepsilon}}\right)^{\varepsilon} \bmod n = a_{i-1}^{\delta_{\varepsilon}\varepsilon} = a_{i-1}^{1+k\phi(n)} = a_{i-1} = P_{LAST}$.

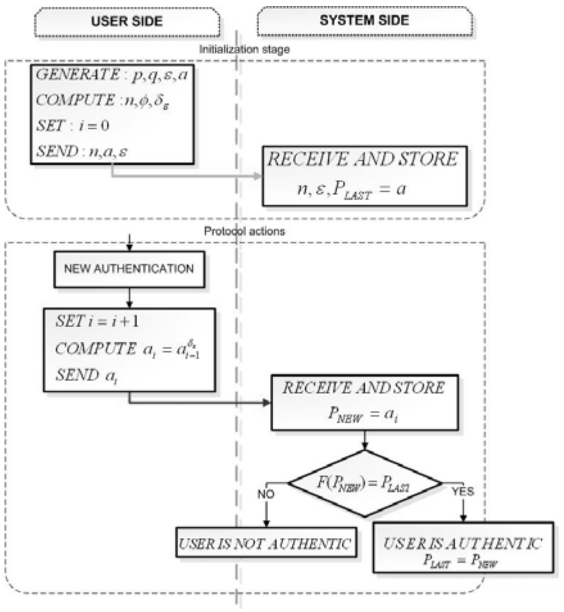
USER SIDE
SYSTEM SIDE
Initialization stage
GENERATE : p,q,ε,a
COMPUTE : $n,\phi,\delta_\varepsilon$
SET : $i=0$
SEND : n,a,ε
RECEIVE AND STORE $n,\varepsilon,P_{LAST}=a$
Protocol actions
NEW AUTHENTICATION
SET $i=i+1$
COMPUTE $a_i=a_{i-1}^{\delta_\varepsilon}$
SEND a_i
RECEIVE AND STORE $P_{NEW}=a_i$
$F(P_{NEW})=P_{LAST}$
NO
YES
USER IS NOT AUTHENTIC
USER IS AUTHENTIC $P_{LAST}=P_{NEW}$

Fig. 3. One-time password authentication scheme with exponentiation over Z_n^* in the particular case $\gcd(\varepsilon,\phi(n))=1$

We will also notice that by setting $\varepsilon=2$ the one-time passwords array $A=\{a,a^2,a^{2^2},a^{2^3},...,a^{2^i},...,a^{2^\eta}\}$ will be a set defined by the relation $a_{i+1}=a_i^2$ which means that the one time password a_i is the quadratic residue of a_{i+1}. Then it is obviously that password a_i may be computed either as a quadratic residue of a_{i+1} either as a^{2^i}. The process of authentication may run just like in the general case with the observation that the next one-time password may also be computed from the one just sent to the system as a quadratic residue if the factorization of n is known.

5. ADVANTAGES AND DISADVANTAGES OF EXPONENTIATION OVER Z_n

In brief, the advantages and disadvantages of the functions proposed will be as follows:
Advantages:
- *Computing the first one-time password is easier.* For big values of η we see that computing F^η is certainly easier since it does not require η compositions of F with herself (we do not have to compute al previous one-time passwords $F^i, i<\eta$). Actually the computation of the last password is not much influenced by η so enormous values of the upper bound B_{U,N_A} may be chosen as well.
- *Knowing the upper bound B_{U,N_A} is not required.* Previous knowledge of the upper bound B_{U,N_A} is not needed for the particular case when $\gcd(\varepsilon,\phi(n))=1$.

Disadvantages:
- *Message size is significantly larger.* A hash function may output results on 128-256 bits while a module should be around 1024-2048 bits, which results in keys that are a few times longer. Big modules are required in order to make the function intractable.
- *Computing all the passwords requires more time with modular exponentiation.* Even if computing the first one-time password could be done faster, the computation of more one-time passwords would require more time for modular functions than for hash functions. Solutions presented in this paper might not be very practical because modular exponentiations are time consuming and each password requires a modular exponentiation. Practically speaking a modular exponentiation might require from tens to thousands of modular multiplications while a modular multiplication could be one hundred times more complex then a hash function. It should be also stated that more advanced authentication techniques such as public-key authentications can be performed with the same amount of computation.

6. CONCLUSIONS

Functions over groups of integer were used instead of hash functions in order to generate one-time passwords. Using these functions has advantages when the number of authentications is high and uncertain by avoiding multiple compositions but also has the disadvantage that they are harder to compute. We expect that algorithms described in this paper are secure and can be used to generate one-time passwords.

ACKNOWLEDGEMENT

The authors thank Dragomir Toma-Leonida and Minea Marius for helpful discussions about this work.

REFERENCES

Chien, Hung-Yu, Jan, Jinn-Ke, (2003). Robust and Simple Authentication Protocol. *Oxford Journal, The Computer Journal*, **Vol. 46**, No. 2, 2003.

Haller, N., (1994). The S/KEY One-Time Password System. RFC 1760, Bellcore.

Haller, N., Metz, C., Nesser, P., Straw, M., (1998). A One-Time Password System. RFC 2289, Bellcore, Kaman Sciences Corporation, Nesser and Nesser Consulting.

Lamport, L. (1981). Password Authentication with Insecure Communication. *Communication of the ACM*, 24, 770-772.

Menezes, A.J., van Oorschot, P.C., Vanstone, S.A., (1996). *Handbook of Applied Cryptography.*

CRC Press.

FIPS 180-1, (1995). National Institute of Standards and Technology (NIST). Announcing the Secure Hash Standard., U.S. Department of Commerce.

Rivest, R., (1992a). The MD4 Message-Digest Algorithm. RFC 1320, MIT and RSA Data Security, Inc.

Rivest, R., (1992b). The MD5 Message-Digest Algorithm. RFC 1321, MIT and RSA Data Security, Inc.

APPENDIX A – NOTATIONS

$\phi(n)$ - Euler ϕ function for an integer n and denotes the number of integers smaller than n and relatively prime to n; in this paper we will use the particular case $n = p \cdot q$ where p and q are prime numbers, then $\phi(n) = (p-1)\cdot(q-1)$ and for any integer $x \in Z_n^*$ holds $x^{\phi(n)} \bmod n = 1$

B_{U,N_A} - the upper bound of the number of authentications

$F^n(x)$ $= \underbrace{F(F(...(F(x))...))}_{n-times}$ - the composition of function $F(x)$ with herself n-times

$\gcd(x, y)$ - greatest common divisor of integers x and y

N_A - the number of authentications allowed

T_A - time required for the authentication process to complete

x^n $= \underbrace{x \cdot x \cdot x \cdot ... \cdot x}_{n-times}$ - the exponentiation of the variable x to n

Z_n $= \{0,1,2,3,...,n-1\}$ - the set of integers modulo n

Z_n^* $= \{x \mid x \in Z_n, \gcd(x,n) = 1\}$ - the set of integers smaller than n and relatively prime to n