A TECHNIQUE FOR REMOTE AUTHENTICATION

William A.Wulf, Alec Yasinsac, Katie S. Oliver, and Ramesh Peri

University of Virginia
Charlottesville, VA 22903

ABSTRACT

Distributed systems have long relied on shared secrets to ensure the authenticity of principals. Public key systems and zero knowledge proofs of identity have reduced this reliance. We offer a method of remote authentication that can be used with no advance shared knowledge by parties, and that allows parties to increase their confidence in the authenticity of a suspicious party to an arbitrary level.

INTRODUCTION

Distributed systems have long relied on shared secrets to ensure the authenticity of principals. Public key systems and zero knowledge proofs of identity have reduced this reliance. We offer a method of remote authentication that can be used with no advance shared knowledge by parties, and that allows parties to increase their confidence in the authenticity of a suspicious party to an arbitrary level.

A central problem in establishing a logically secure channel for communication between distributed users lies in "guaranteeing" the identity of the two participants, known in the literature as authentication [1], [2]. Traditionally, authentication between two principals has relied upon each knowing some information that only the two participants know and that cannot be easily guessed, i.e., a password. In cryptographic systems, this password has been transformed into a "secret key" and is now used both as the foundation for the authentication of principals and for the privacy of messages. Requiring principals to share a password or other piece of secret information as the basis for authentication poses many potential problems such as vulnerabilities to guessing attacks and replay attacks [3], [4].

Another problem with shared knowledge, or secret keys, is that with a large number of principals connected in a distributed system, it may be impractical for each principal to have a unique secret key relationship with every other principal they desire to communicate with. A reasonable upper bound on the total number of keys required is N^2 where N is the number of principals on the network. Managing (creating, distributing and storing) these secret keys constitutes a significant security risk. One common method for resolving this issue has been to introduce a centralized "trusted agent" or Authentication Server (AS) [1], [2], [4], [5]. Unfortunately, a centralized AS also introduces a single point of failure and potential bottleneck into the system.

With or without a centralized AS, key management is very complex. For example, when adding a node to the distributed system, normally, no secure channel exists initially between the new participant and any other participant. Nonetheless, we would like to be able to authenticate new remote users over the net. Public key cryptography, first described in [6], circumvents the first time authentication problem by allowing a participant to broadcast their public key and later use the inverse key, which cannot be easily compromised, to authenticate themselves [7], [8], [9]. Unfortunately, an integrity problem exists with the initial distribution of the public key. That is, anyone can publish a public key and claim any identity they desire without some other authentication method in place. What public key authentication ensures is that the participant that published the public key is the participant that later authenticated with the public key's inverse. Though not a perfect solution, this is a step in the right direction. The mechanism we offer in the next section attacks the problem of the initial key distribution similarly to public key methods, but has strengths not shared by public key systems.

It is generally recognized that passwords and keys should be changed regularly. If a secure channel exists, it would be convenient to utilize that channel to pass new keys. Unfortunately, basing the security of future channels upon the security of a current key conflicts with the time problem just mentioned by making the compromise of a key perpetual. Even if an intruder takes a long time to compromise a key, once it is broken, they can merely "run up the chain" of keys to the current one, quite possibly gaining some important information along the way. This problem is the subject of much research and many mechanisms have been proposed to resolve it [1], [4], [5]. One way to address both the prob-

lems of chaining and replay, is to devise a mechanism for authentication that is independent of all previous keys and of all previously shared secrets. A mechanism that allows the verifier to ask a question to which the answer can be verified without knowing what computations lead to the answer would have this property.

This paper introduces such a mechanism. The following sections offer the foundation for our method, identify important issues in selecting functions that allow the method to perform as we desire, propose a specific algorithm for accomplishing authentication in a distributed environment, and conclude with a summary and discussion of future work.

AUTHENTICATION WITHOUT SHARED KNOWLEDGE

While classic authentication depends upon participants sharing some secret, our approach to authentication is to formulate a mechanism that does not require passing any shared secrets across the channel. Suppose a client A requests a set of services from another principal B. In order to authenticate user A, B would like to ask a question that only A (not even B) knows the answer to. However, B must be able to verify the answer received.

Consider two functions f and g, with the following two properties:

f & g are hard to invert, and
$g(f(x),f(y)) = f(g(x,y))$.

When user A signs on, she selects an appropriate f that only A knows. This function f will be used to identify user A to B. A then announces herself by providing B with an arbitrary value x_0 along with $f(x_0)$ in the clear.

Step 1. A -> B: A, x_0, $f(x_0)$.

B will use this x_0, $f(x_0)$ pair along with a function g selected by B to authenticate future requests for service from A. Only B will know g. In order to authenticate this user, B would generate a random number, call it y, and send it to the user along with $g(x_0,y)$. The user would return f(y) and $f(g(x_0,y))$.

Step 2. A -> B: A
Step 3. B -> A: y, $g(x_0,y)$
Step 4. A -> B: f(y), $f(g(x_0,y))$.

B would then calculate $g(f(x_0),f(y))$ and compare that value against $f(g(x_0,y))$ supplied by A in step 3. If the values are the same, B can conclude, with probability 1/m, where m is the size of the range of f, that the user is who she claims to be. If a greater confidence level is desired, B can repeat the process with a different y and corresponding $g(x_0,y)$ value to increase the probability to $1/m^2$, repeat again for $1/m^3$, etc.

There are two critical elements to this mechanism. First, an intruder must not be able to predict $g(f(x_0),f(y))$, so g must be known only to B. If g should become public, an intruder that has been monitoring the system can know the values of $g(x,y) = c_1$, $f(x) = c_2$, and $y = c_3$. In order to masquerade as A, the intruder needs only compute:

$$f(g(x,y)) = g(f(x),f(y)) \text{ or } f(c_1) = g(c_2, f(c_3)).$$

Given these values, the intruder can select an arbitrary value for $f(c_3)$, call it c_4, and apply g using the constants c_2 and c_4 to it and produce a result, call it c_5. By definition, c_4 and c_5 will have the property defined by the authentication functions f and g, and can be used by an intruder to fool an authenticator.

The second requirement is the inability of an intruder to compute f(y) and f(g(x,y)). Only the user can know f. Should one decide to make f public, or if f is compromised, an intruder could easily masquerade herself as that user. By knowing f, the intruder can easily compute f(y) and f(g(x,y)) correctly and pass herself off as another user.

Neither f nor g may be known by anyone other than the two involved parties, and neither is shared. Even though we cannot make the functions f and g public, it is feasible to make the forms of f and g public. This way each user can choose an appropriate function for f in private, not requiring passage of f over any channel or verification by any centralized Authentication Server. To attack the protocol messages, a potential intruder must attempt to invert the functions f and g. For this reason, we believe this method is as strong as any public key method yet known.

Relationship to Public Key Cryptography

It is noted that this method shares the integrity problem of public key systems, in that it does not provide absolute authentication without prior shared secrets. It does provide compelling evidence that a user announced as user [A, x, f(x)] is the same user requesting the current service, without shared secrets. If the announced values we describe are verified between parties a priori, as pub-

lic keys must be, this mechanism can be used similarly to a public key authentication system.

While this method shares other similarities with public key authentication mechanisms, it differs significantly in its purpose. Public key methods are related to secret key methods in that both perform the dual functions of providing data privacy and allowing authentication of principals. Though these roles could be independent, they are "coupled" by public and secret key mechanisms in that both are accomplished by encryption of data. A result of this is that encryption based mechanisms base authentication on the subtle rule: "If two principals A and B share a secret key k [or k and k^{-1}], and A sees a message M encrypted under k [or k^{-1}], then A is justified in believing that B sent M" [paraphrased from [2], p21]. On the other hand, our method does not depend on encryption. No keys are generated and all values are passed in the clear since there is no concern for privacy. The purpose of our method is strictly for authentication, so there is no need for such subtle assumptions.

Another strength of our method is its inherent distributed nature. Many encryption based authentication systems rely on centralized authentication or name server support, introducing the classic problems of single point of failure and system bottleneck into the cryptographic system. Our method requires no centralized support and is, thus, well suited to a fully distributed environment.

A third distinction between our method and public key methods relates to ability of the authenticator to vary the nature of the authentication sequence. Public key systems give authenticators no flexibility in varying the identifying information of the parties desiring authentication. The requestor publishes a public key and the authenticator must use it as the identifier. Only the requestor can change this identifier, and changing a public key is a sensitive operation. Our method allows the authenticator to change the "identifier", i.e. the x_0,$f(x_0)$ pair, of a remote principal. The authenticator must use the original x_0,$f(x_0)$ pair in the first authentication run, but in subsequent runs, the authenticator may replace the x_0 and $f(x_0)$ by the y and f(y) pair collected in the first session, as follows:

Step 5. A -> B: A
Step 6. B -> A: y', g(y,y')
Step 7. A -> B: f(y'), f(g(y,y')).

The g(x0,y) and f(g(x0,y)) pair may also be used to replace a previous identifier pair. Since the authenticator selects the value of y, the new identification pairs are unpredictable (to the degree of strength of the uninvertable function) by potential intruders. This flexibility gives the authenticator the ability to vary subsequent authentication sequences, which ensures their independence. This inherently defends against replay attacks, which public key schemes must address with mechanisms in the protocol.

The authenticator can force further variations in the interaction by changing the function g they use for authentication. This function is under the total control of the authenticator. As long as the new function g meets the prescribed form, the authenticator can change g at any time with no coordination required and authentication can continue as described. The flexibility the authenticator has in varying the interactions clearly distinguish this method from public key systems.

Also note that unlike public key schemes, our method has no key. The length of the key serves as an upper bound on the key space, so given enough time and computing power, an intruder could theoretically compromise a key by brute force. There is no corresponding upper bound for the identifiers or functions used for authentication in our method. The number of pairs of identifiers (x, f(x)) and number of functions available as f and g are unbounded when carefully selected.

Relationship to Zero Knowledge Proofs

This mechanism is also similar to Zero Knowledge Proofs (ZKP) first published by Goldwasser et al. in [10] and other works. ZKP's have the remarkable property of being both convincing and yielding nothing except that the assertion is indeed valid [11]. The protocol described in this paper is Zero Knowledge since an intruder who uses a recorded message can only play back the recorded message if the questions asked happen to be the same. An intruder who records many authentication processes cannot increase his chance of masquerading herself as another user since the questions asked are completely random. In ZKP's, a prover A wants to prove that they know the answer to a question posed by an identifier without revealing the answer. In our mechanism, the prover or party being authenticated passes the answer to the questions asked across the net in the clear. The trick is that the verifier can ensure that the correct answer is passed because they know the form of the question (i.e. the equations).

In 1986, Fiat and Shamir presented a zero-knowledge identification scheme which enables any user to prove his identity without shared or public keys [12]. Their scheme requires users to carry a "smart card" as proof of

identity. A trusted center is required to issue these smart cards, however no further contact with the center is required. Each smart card contains the user's unique identifier and k other values used in establishing identity. When a user wished to identify herself, the verifier must prove that it knows these k values without giving away any information about their values. For specific implementation details of the protocol, the reader is referred to [12].

Fiat and Shamir's protocol is similar to the one described in this paper in that the verifier has gained no useful information that can later be used to masquerade as that user. Fiat and Shamir choose to make their functions public while keeping the user's identity values private and stored on a smart card. The smart card required by Fiat and Shamir, only obtainable from the trusted center, can easily be lost or stolen and therefore compromised. The protocol described here, on the other hand, keeps functions private, while allowing the user's identity values to remain public. Our method allows the user to choose his identifying function in complete secrecy and only requires a calculator to compute the values requested by the protocol. If a user at a workstation wishes to authenticate herself when requesting a service, our method can be easily automated and stored on the workstation.

SELECTING SUITABLE FUNCTIONS

When selecting the actual functions for f and g, there are two properties we must keep in mind. The first requirement is that f must distribute across g so that the relation $f(g(x,y)) = g(f(x),f(y))$ holds. For example, if we choose the simple functions $f(x) = x^2$, and $g(x,y)= x*y$, we see that:

$$f(g(x,y)) = x^2y^2$$
$$g(f(x),f(y)) = x^2y^2.$$

This is a nice example that demonstrates the relation, and it would be convenient to use these particular functions for f and g, since each user could simply pick different exponents and the relation with g would still hold. Unfortunately these functions are easy to invert. It is clear that we must select more sophisticated functions for f and g. It is easy to find complicated, hard to invert functions, however many of the operations that make functions one-way also fail to exhibit the distributive property that enables the described scheme to work.

So our challenge has become to find functions that distribute over each other (or one that distributes over itself) and both are one-way with any function that distributes over g also one-way. There are many well-known functions that have been designed to be one-way. DES is one such function that was specifically designed to be hard to invert [13]. However, whether DES distributes across some operation in a way that satisfies our algorithm is an open question.

The Discrete Exponentiation Function

Discrete exponentiation is a well-known one-way function. That is, given a value f(x) for a discrete exponentiation function f, there is no known method to easily (in polynomial time) derive the value of x. When combined with multiplication we have an example of operations that distribute nicely over one another and, create a one-way function. If we select

$$f(x) = x^a \bmod n,$$
$$g(x,y) = (xy)^b \bmod n$$

then we can show that f and g have the commutative property we desire as follows:

$$\begin{aligned} g(f(x),f(y)) &= ((x^a \bmod n) * (y^a \bmod n))^b \bmod n \\ &= ((x^a - k_1n) * (y^a - k_2n))^b \bmod n \\ &= ((xy)^a - k_1ny^a - k_2nx^a + k_1k_2n^2)^b \bmod n \\ &= (xy)^{ab} \bmod n \end{aligned}$$

and

$$\begin{aligned} f(g(x,y)) &= ((x * y)^b \bmod n)^a \bmod n \\ &= ((x * y)^b - k_3n)^a \bmod n \\ &= (xy)^{ab} \bmod n. \end{aligned}$$

AUTHENTICATION USING F AND G

For the purpose of illustration, we construct a protocol to implement the described technique with a one-way discrete exponentiation function. Choose n, a, and b large primes, and f and g are as defined above. n, the form of g, and the form of f will be public knowledge; the specific functions f and g, and values a and b are kept secret. The server will also select a random y for use in the second step of the authentication sequence. Upon login, the requestor A will select a sufficiently large x and will announce herself to B with the message:

A -> B: A, x, $x_a \bmod n$

When the requestor desires service, she will so indicate to B and the authentication sequence will progress as follows:

A -> B: A
B -> A: y, $(xy)^b \bmod n$
A -> B: $y^a \bmod n$, $((xy)^b \bmod n)^a \bmod n$)

B would then verify the authenticity by computing:

$((x^a \bmod n)(y^a \bmod n))^b \bmod n = ((xy)^b \bmod n)^a \bmod n$)

If the equivalency holds, then B can be confident, though not certain, of the identity of the requestor. Additional iterations of the authentication sequence increase the probability of actual authentication exponentially. In this way, the proposed method is similar to the Zero Knowledge proofs proposed by Goldwasser et al [14]. From B's viewpoint, the ability to achieve an arbitrary degree of confidence in the identity of a user could be highly useful. In order to ensure security of the method, g could be changed with virtually every iteration of the authentication sequence.

Security of this method lies in guaranteeing the privacy and unpredictability of f and g. Several characteristics of this protocol support the proposition that predicting either of these functions will be very hard. First, the low number of computations passed across the network for each specific f and g give the cryptanalyst little data to work with. Each activation of the authentication sequence passes only three known plain text pairs across the wire. What is more, the cryptanalysis or detection of a correct f or g is very hard to verify. The g function may be changed with virtually every iteration of the authentication sequence, and the f function will change with each login. By the time the cryptanalyst has proposed a function, the function may have changed. Even if not, an authenticator could easily keep track of erroneous authentication sequences to limit the number of guesses an intruder could make to find an accurate f as described in [3].

CONCLUSIONS AND FUTURE WORK

We have presented a method for one-way authentication of users in a distributed system which requires participants to share only the forms of the functions f and g and the prime number n used in our example. A user can select a specific function f in complete secrecy. Similarly, a server can select it's function g and keep it a complete secret from the rest of the world. The only requirement is that the functions must match the specified published forms. This allows the desired relationship between f and g to hold.

This method also does not require any messages sent between parties be encrypted, rather everything is sent in the clear. We are able to send these messages in the clear for two reasons. First, the participant being authenticated is being asked a question that only that participant knows the answer to. An intruder monitoring the network can see the question go by, and even see the answer to that question. Since we are using one-way functions as the basis for our questions, the next time a question is posed, still no one but the user can produce the correct answer to the question. Secondly, in the sense that the functional values of the challenges are similar to encrypted data, those values are "non-verifiable plaintext" as described in [3]. Since there is no intuitive meaning associated with the values selected, the only way an intruder can verify them is by utilizing the function.

This method can verify the authenticity of a user to an arbitrary degree of confidence. If the probability of an intruder guessing appropriate values for the authentication sequence is 1/m, where m is the size of the range of f, then repeating the authentication sequence will give the authenticator the confidence that the probability that the intruder can guess the correct values is $1/m^2$. In this way, our method is similar to the "Zero Knowledge" paradigm. However, this method does not require any prior shared secrets and still allows a user to arbitrarily announce herself. Based on the announcement, the authenticator can be guaranteed that the user requesting authentication, is the same user who announced themselves at login.

Future work focuses on finding additional one-way functions that distribute across each other. We have looked at the properties of several known one-way functions and found that most cannot be applied to our algorithm. We would like to know if the DES algorithm has any properties that we could apply to our f and g functions. We speculate that we will have to find some function that distributes across one of the known "hard to invert" functions from number theory to use in the f and g functions.

Since this is a new protocol, fundamentally different from more classical protocols, we are interested in finding methods that can be used to verify whether the protocol meets its goals. We would like to know if and how existing verification methods such as BAN Logic and Interrogator can be applied to this protocol that does not depend on shared secrets [2], [15].

REFERENCES

1. Needham, R. M., Schroeder, M. D. “Using encryption for authentication in large networks of computers”. Communications of the ACM, VolV 21, No. 12, Dec 1978, 993-999.
2. Burrows, M., Abadi, M., Needham, R. M, “A Logic of Authentication,” ACM Transactions on Computer Systems, Vol 8, No. 1, Feb 1990, 18-36.
3. Lomas, M., Gong, L., Saltzer, J. H.Roger Needham, “Reducing Risks from Poorly Chosen Keys”, Operating Systems Review, 12th ACM Symposium on Operating Systems Principles, Vol 23, Number 5, 3-6, Dec 1989, 14-18.
4. D. E. Denning and G. M. Sacco, “Timestamps in key distribution protocols,” Communications of the ACM, Vol 24, no. 8, Aug 1981, 33-536.
5. Miller, S.P., B.C. Neumann, J.I. Schiller, & J.H. Saltzer, “Kerberos Authentication and Authorization System”, Project Athena Technical Plan, Section E.2.1, MIT.
6. Diffie, W., and Hellman, M., “New directions in cryptography,” IEEE Transactions on Information Theory IT-22, 11(Nov. 1976), 644-654.
7. Rivest, R. L., Shamir, A., and Adleman, L. “A Method for Obtaining Digital SIgnatures and Public-Key Cryptosystems”. Communications of the ACM, Vol 21, No. 2, Feb 1978, 120-126.
8. Nechvatal, James, National Institute of Standards and Technology Special Publication 800-2, “Public-Key Cryptography,” Apr 199,11-11.
9. Luciano Dennis, Prichett, Gordon, “Cryptology: From Caesar Ciphers to Public-key Cryptosystems”, The College Mathematics Journal, Vol 18, Jan 1987, 2-17.
10. Shafi Goldwasser, Silvio Micali,and Charles Rackoff, “The Knowledge Complexity of Interactive Proof Systems,” Proc. 27th Annual IEEE Symposium on Foundations of Computer Science, 1985, 291-304.
11. Oded Goldreich, Silvio Micali, Avi Wigderson, “Proofs that Yield Nothing But their Validity and a Methodology of Cryptographic Protocol Design,” IEEE, 1986, 174-187.
12. “How To Prove Yourself: Practical Solutions to Identification and Signature Problems,” Advances in Cryptology - Crypto ‘86, 186-194.
13. “Data Encryption Standard,” FIPS PUB 46, National Technology Information Service, Springfiled, Va, 1977.
14. Goldwasser, Shafi, Micali, Silvio, and Rackoff, Charles, “The Knowledge Complexity of Interactive Proof Systems,” Siam Journal of Computing, Vol 18, No 1, Feb 1989, 186-208.
15. Millen, J.K., Clark, S. C., and Freedman, S. B. “The interrogator: Protocol security analysis,” IEEE Transactions on Software Engineering, SE-13, 2, Feb 1987, 274-288.
16. McEliece, R.l J. “A public-key cryptosystem based on algebraic coding theory”, DSN Progress Report 42-44, Jet Propulsion Lab., Jan and Feb. 1978.
17. Merkle, R., and Hellman, M. E., “Hiding information and receipts in trapdoor knapsacks,” IEEE Transactions on Information Theory IT-24, Sept 1978.
18. R. C. Merkle, “Secrecy, Authentication, and Public Key Systems,” An Arbor: UMI Research Press, 1982.