International Journal of Computer Trends and Technology
ISSN: 2231-2803/ https://doi.org/10.14445/22312803/IJCTT-V68I1P111

Volume 68 Issue 1, 45-51, January 2020

Original Article

An Enhanced Mutual Authentication Scheme using One-Time Passwords with Images

P. S. V. Vachaspati[1], P. Pardhasaradhi[2], SK. Nazeer[3]

[1,2,3] *Professor & Department of CSE Bapatla Engineering College, GBC Road, Bapatla, Guntur-Dist, Andhra Pradesh, India.*

Received Date: 21 November 2019
Revised Date: 17 January 2020
Accepted Date: 26 January 2020

Abstract *- Is it possible to prove a statement without yielding anything beyond its validity? Zero-knowledge protocols, as their name indicates, are cryptographic protocols that do not reveal any information or secret themselves during the protocol or to any eavesdropper. These proofs possess some very interesting features, and as the secret itself is not transferred to the verifying principal, they cannot try to masquerade as you to any third party*

Keywords *- Zero-knowledge, Identity authentication, Visual Cryptography, Mutual Authentication*.

I. INTRODUCTION TO PROOF CONSTRUCTION FOR ZERO-KNOWLEDGE

It is argued that the notion of proof in a zero-knowledge protocol is somewhat different from the concept of proof in a strict mathematical sense and more similar to a dynamical way of understanding the proof by interaction and interpretation used by humans, i.e. trust will be built slowly. Mathematical proofs are more rigid; they contain a static and formal nature as they are either self-evident or are obtained from previous rules. However, humans' beings tend to use a more intuitive sense of proof where the correctness of a statement is established through the process. In a similar way, in a zero-knowledge proof protocol, instead of providing a static proof for the claim, the prover (one principal) tries to convince the verifier (the other principal) by interactively convincing him of the proof. This type of dynamic trust-building (at least in a weak sense) is absolutely necessary to the non-triviality of the notion of zero-knowledge proof.

One mathematical model is Lagrange's equation for secret sharing. Another technique is visual cryptography, where the user will use his eyes to authenticate himself sharing a secret. We use these things and later go to a novel proposal for authentication [1][2].

Image dividing is a subset of secret sharing because it behaves as a prominent approach to the general secret sharing problem. The secret, in this case, is concealed in images. Without the problem of inverse computations, the secrets may be interpreted correctly to reveal the true meaning of the secret. Image sharing defines a scheme that is identical to that of general secret sharing. In (k,n) image sharing, the image that carries the secret is split up into n parts (known as shares), and the decryption will totally fail unless at least k shares are superimposed. Visual cryptography was originally invented by Moni Naor and Adi Shamir in 1994 at the Eurocrypt conference [5].

Shamir [3] generalized and formalized the definition of the (k,n) threshold scheme. The definition can be presented as follows: Let D be the secret to be divided among n involved parties. A (k,n) threshold scheme is a way to distribute D into n pieces d_1,d_2,…,d_n that satisfies the conditions given here:

- Awareness of any k or more D_i pieces makes the D easily obtainable,
- Awareness of any k-1 (or less) number of D_i pieces leaves D completely unobtainable (in a way that all its possible values are equally likely).

The individual image shares give no clue no matter how much computational power is available to the attacker. In the display of the secret image, the two shares that are generated are required for the recovery of the secret [6][7].

An active adversary (perhaps by co-working with his friends distributed over an open communication network) is capable of intercepting, modifying, or injecting messages and is good at doing so by impersonating other protocol principles. Even in the existence of active adversaries and communication errors, a secure cryptographic protocol should meet all claimed objectives. A successful attack on an authentication protocol usually does not refer to breaking a cryptographic algorithm, e.g., via complexity theory-based cryptanalysis technique. Instead, it usually refers to the adversary's unauthorized and undetected acquisition of cryptographic credentials or nullification of cryptographic service without breaking a cryptographic algorithm. Of course, this is due to an error in protocol design, not the one in the cryptographic algorithm [4].

Classical protocols for the identification of principals in a transaction suffer from flaws that are inherent to the process used to achieve the objective. In simple password protocols, claimant A gives his password to a verifier B. If some precautions are not taken, an eavesdropper can get hold of the password that was transferred, and afterwards, he can impersonate as A. Other protocols try to improve on this, as in the case of challenge-response systems. In this type of protocol, A accepts B's Challenge to provide proof of knowledge of a shared secret. Of course, the Challenge is changed every time the protocol is used; therefore, an eavesdropper can, in time, gather enough partial information about the shared secret to try an impersonation attack like the one described above. Here we will discuss Zero-Knowledge Protocols (abbreviated ZKP from here on), which are designed to defeat the disadvantages described above. In ZKP, a prover will try to exhibit knowledge of a certain secret to a verifier. The idea is to allow the proof to take place without revealing any information whatsoever about the proof itself, except, of course, for the fact that it is indeed a valid one. Zero-Knowledge Proofs can be compared to an answer obtained from a trustworthy oracle. It must be mentioned that the concept of proof in ZKP is different from the traditional Mathematical concept. Mathematical proofs are strong, using either self-evident statements or statements obtained from proofs established in advance.

ZK proofs are more like the dynamic process used by human beings to establish the truth of a statement throughout the exchange of information. Furthermore, in a ZKP, instead of presenting a static proof for a statement, the proving principle involves the verifier in a process in which he tries to satisfy the verifier of the truth of the statement interactively.

A. Features

Zero-Knowledge Protocols have the properties given below [95, 96]. Suppose A and B (prover) be principals:

- The verifier cannot learn anything from the protocol. The verifier does not get anything extra in the process of the proof that he could obtain from public information by himself. This is the central idea of zero-knowledge, i.e., zero amount of knowledge is transferred. There are several similar protocols, called Minimum Disclosure Protocols, which relax this property, trying to maintain the flow of information to a minimum extent.
- The proving principle cannot deceive the verifier. If B doesn't know the secret, he can only fool A with an incredible amount of luck which nobody possesses. The odds that an impostor can deceive the verifier can be made as low as necessary by making more the number of rounds executed in the protocol.
- The verifying principal cannot cheat the prover. A can't get any information out of the protocol, even if she doesn't stick to the rules. The only thing A can do is decide whether she accepts that Pat actually knows the secret. The proof provider will always reveal one solution of many; by doing this, he ensures that the secret remains with him.
- The verifier cannot pretend to be the prover to a third party. As stated earlier, no information flows from B to A. This precludes A from trying to masquerade as B to a third party. Worthless in convincing a third party

To obtain a definition of the Zero-Knowledge Protocol, we will discuss a few properties that must be satisfied in order to ensure the behaviour promised for such systems. These properties include soundness and completeness in the context of Interactive Proof Systems.

II. INTERACTIVE PROOF SYSTEMS

Zero-Knowledge Protocols are the instances of Interactive Proof Systems, where a prover and a verifier shift challenges and responses, dependent on random numbers (the outcomes of fair coin tosses in general), which they are allowed to keep secret. As we said above, the proof in this context is probabilistic rather than absolute as in the mathematical sense. These proofs need only be correct with a certain bounded probability (of course, this probability can be made arbitrarily close to 1). Interactive proofs are sometimes called proofs by protocol. Interactive proofs used for identification may be treated as proofs of knowledge. Suppose P has a secret s, and he wishes to convince V that he has knowledge of s by responding correctly to queries of V (such queries involve publicly known inputs and agreed upon functions) that require knowledge of s to answer. Note that it is quite different to prove knowledge of s than it is to prove the existence of s. For example, proving that a certain x is a quadratic residue modulo n differs from proving knowledge of the square root of x modulo n.

We may say the interactive proof is said to be proof of knowledge if it has the properties of soundness and completeness. These properties are defined here.

A. Completeness Property

An interactive proof protocol is complete if, given an honest prover and an honest verifier, the protocol succeeds with overwhelming probability (i.e., the verifier accepts the prover's claim). The definition of overwhelming, of course, depends on the application but generally implies that the probability of failure is not of practical significance.

B. Soundness Property

An interactive proof protocol is sound if there exists an expected polynomial-time algorithm M with the following property: if a dishonest prover (impersonating B) can with non-negligible probability execute the protocol with A, then M can be used to extract from this prover the knowledge (essentially equivalent to B's secret) which with overwhelming probability allows subsequent protocol executions. Since any party capable of impersonating B must, in fact, have knowledge equivalent to the secret itself, the soundness property guarantees that the protocol is, in fact, providing proof of knowledge (in order to succeed, you must count on knowledge equivalent to the secret). This property, therefore, prevents a dishonest prover from

succeeding. A standard technique used to prove that a certain protocol is sound is to assume the existence of a dishonest prover that is capable of successfully executing the protocol and show how this allows computing the secret in polynomial time. This "proof of knowledge" idea is the foundation of zero-knowledge proofs. But, it is clear that neither of these properties says anything about zero-knowledge itself. A ZKP should, in addition, have the property that no amount of knowledge should pass between the prover and the verifier that the verifier couldn't conclude without the help of the prover. This property is simply called the zero-knowledge property and will be defined after introducing the concept of a simulator, as discussed below.

C. Zero-Knowledge Property

Proof of knowledge has the zero-knowledge property if there exists a simulator for the proof. This formalizes what has been said in previous sections. In the context of a ZKP, the verifier does not obtain further information about the secret other than its validity. Furthermore, a much-desired property of this type of protocol is that the number of times that the prover participates in them does not vary the chances of success of impersonation attacks (as it might in password or challenge-response protocols). The Zero-Knowledge Property allows us to arrive at the definition of Zero-Knowledge Proof as follows.

D. Zero-Knowledge Proof

A Zero-Knowledge proof is a proof of knowledge that also has the Zero-Knowledge property.

III. OBSERVATION

It is clear that the Zero-Knowledge property and the soundness property have no say in the level of security that a system presents. It is of key importance to the security of a given protocol for it to depend on computationally difficult problems. No proofs exist for the most commonly used problems (e.g., integer factorization, knapsack problem, discrete logarithm, etc.), so the security of the systems that use them are directly dependent on future developments in the field of Computational Complexity. This type of system is commonly referred to as provably secure. A few points can be made in the difference between Zero-Knowledge and public key (PK) techniques. These are:

- No degradation with usage: Repeated use of a ZK protocol does not present degradation. ZK protocols are also resistant to chosen text attacks. This leads a ZK protocol that is not provably secure to be considered against a PKP which is provably secure.
- Unproven assumptions: Most ZK and PK protocols depend on the same assumptions (quadratic residuosity, factoring, discrete log, etc.).

Considering the above discussion as a base, we can enhance the protocol. The protocol which we have described in the previous chapter can be converted to a Zero-Knowledge protocol. Here we give the enhancement. Here the verifier challenges the Claimant a number of times, and each time the Claimant is supposed to accept the Challenge.

If this is done successfully, the verifier builds enhanced trust in Claimant. Similarly, the Claimant challenges the verifier several times and builds trust, which is mutual authentication. Here each time, the password is not the same but quite different and unpredictable for anyone. Here the two principals are building trust in each other.

A. Authentication based on trusted freshness

Here we will introduce the security definitions based on trusted freshness. Let's introduce some notations related to trusted freshness security analysis.

- Principals, probabilistic polynomial-time machines, which are interconnected by point-to-point links over which messages can be exchanged.
- Freshness identifier (or TVP), a unique freshness component generated for a particular protocol run, can be a nonce, a timestamp, a session key or a shared part of a session key.
- Protocol, a communication procedure that is run between or among co-operative principals.
- Message-driven protocols, protocols are initially triggered at a party by an external "call" and later by the arrival of messages.
- Challenge-Response protocol, in a challenge-response mechanism, one participant can verify the lively correspondence of the intended opposite partner by inputting a freshness identifier (Challenge) to a composition of a protocol message, and the composition involves a cryptographic operation (response) performed by the intended opposite partner.
- The session, a copy of a protocol run at a party, several copies of any protocol may be simultaneously run by each party.

B. Trusted freshness

In the context of communication protocols, that a freshness identifier is fresh means that the identifier has not been used previously and originated within an acceptably recent time. Formally, fresh typically means recent, and it is in the sense of having originated subsequent to the beginning of the current protocol instance. Note that such freshness alone does not rule out interleaving attacks using parallel sessions.

C. Freshness

Given a protocol Π between partners A and B. A component of the protocol Π is fresh if the component is guaranteed to be new from the viewpoint of one party A or B.

Classification of freshness component

Three freshness component categories are classified depending on the type of the component:

a) Time-stamp Recording the time of creation, transmission, receipt or existence of information. The party originating a message obtains a timestamp from its local

(host) clock and binds it to a message. Upon receiving a timestamped message, the second party obtains the current time from its own (host) clock and subtracts the timestamp received. The timestamp difference should be within the acceptance window, and each party should maintain a "loosely synchronized" clock which must be appropriate to accommodate the acceptance window used.

The time clock must be secure to prevent adversarial resetting of a clock backwards so as to restore the validity of old messages or set a clock forward to prepare a message for some future point in time.

b) Nonce A value originated subsequent to the beginning of the current protocol instance, and it is used no more than once for the same purpose. In a challenge-response protocol, the term nonce is most often used to refer to a "random" number that is sampled from a sufficiently large space, but the required randomness properties vary. A key parameter or the shared parts of a key may be viewed as a nonce in some cases. If random numbers are chosen by A and B, respectively, then the random numbers together with a signature may provide a guarantee of freshness and entity authentication.

c) Sequence number A value provided by a never-repeated sequential counter, and it serves as a unique number identifying a message and is typically used to detect message replay. A message is accepted only if the sequence number therein has not been used previously (or not used previously within a specified time period). Sequence number changes on every new protocol instance or new message depending on different purposes. Each party should maintain the sequence number pairwise of the originator and the receiver and be sufficient to determine previously used and/or still valid sequence numbers. Distinct sequences are customarily necessary for messages from A to B and from B to A. Sequence numbers may provide uniqueness but not (real-time) timeliness and thus are more appropriate to detect message replay than entity authentication. Sequence numbers may also be used to detect the deletion of entire messages; they thus allow data integrity to be checked over an ongoing sequence of messages, in addition to individual messages. Note that a sequence number is not natively fresh, even for the sequence number generator. The cost for a random number or a sequence number to provide a freshness guarantee is an additional message more than that for the timestamp in the one-pass technique.

IV. CREDIBLE RECENTNESS

Given a protocol Π between partners A and B in the presence of a probabilistic polynomial-time adversary I that has full control of the communication links. We say that the freshness identifier γ is fresh, or in other words, a trusted freshness, if for a participant A,

1) The freshness identifier γ originates in participant A itself.

2) The freshness identifier γ is a timestamp, and the timestamp difference between the initiator and the receiver is within the acceptance window.

3) A has corroborative evidence that γ is fresh. Here the corroborative evidence may be a signature, a MAC or other one-way transformation, including the freshness identifier.

The first sufficient condition is based on the randomization of A's nonce, which has been sampled at random from a sufficiently large space, and so no one can predicate the value before sampling; the second sufficient condition is based on a "loosely synchronized" clock, and the timestamp difference is within the acceptance window. Hence the "recent" property could be checked by the opponent party; The third sufficient condition is based on the security property of the cryptographic algorithms, and it is widely used and more useful in challenge-response protocols. Note that the freshness of a freshness identifier could be given via mathematical proofs.

A. Generation Rule

Given a protocol Π between partners A and B in the presence of a probabilistic polynomial-time adversary I that has full control of the communication links. If a freshness identifier γ is a nonce or a timestamp, and it is generated by participant A itself, then A believes that γ is a trusted freshness.

Rationality, A freshness identifier γ, originating in the participant A could be a nonce, a timestamp or a sequence number. Let's consider the freshness of γ in these three cases:

a) The freshness identifier γ is a nonce.

Let's recall the supposition: the term nonce is most often used to refer to a "random" number that is sampled from a sufficiently large space. The "random" numbers are, in fact, pseudo-random numbers, they are generated by a pseudo-random number generator, and they have a distribution totally determined (i.e., in a deterministic fashion) by a seed. Yet, a good pseudo-random number generator yields pseudo-random numbers, which are polynomials indistinguishable from truly random numbers. Recall that adversary I am a probabilistic polynomial-time machine that has full control of the communication links. Hence, the adversary I couldn't distinguish the distribution of the output of the random variable from a pseudo-random number generator from the uniform distribution of strings (truly random numbers) which are of the same length as those of the pseudo-random variables. Hence, if a freshness identifier γ is generated by participant A itself, then A believes that γ is recent and it is a "random" number. Since γ is a "random" number, it couldn't be guessed by a probabilistic polynomial-time attacker I, and we can guarantee that the freshness identifier γ is used no more than once for the same purpose. That is, the freshness identifier γ is a trusted freshness.

b) The freshness identifier γ is a timestamp.

Since the freshness identifier γ is generated by participant A itself, then A believes that γ is recent. Recall the supposition that the timestamp difference between the initiator and the receiver is within the acceptance window, so there is no time gap for the freshness identifier γ to be used for other purposes. That is, the freshness identifier γ is used no more than once for the same purpose. Hence the timestamp γ is a trusted freshness.

c) The freshness identifier γ is a sequence number.

The sequence number is a value provided by a never repeated sequential counter, and it is typically used to detect message replay. Since the freshness identifier γ is generated by participant A itself, A believes that γ is recent. However, the sequence number γ may be guessed even by a probabilistic polynomial-time attacker I, so I could obtain the intending response messages from sending request messages to the victim oracle, and the attacker I may replay the achieved messages including γ for other purposes. Hence, we do not regard a sequence number as a trusted freshness.

B. Timestamp Rule

Given a protocol Π between partners A and B in the presence of a probabilistic polynomial-time adversary I that has full control of the communication links. If a freshness identifier γ is a timestamp, and it is received by A from opponent B, then A believes that γ is a trusted freshness.

a) Rationality: Since the freshness identifier γ is a timestamp, and the timestamp difference between the initiator and the receiver is within the acceptance window, so A believes that γ is recent and there is no time gap for the freshness identifier γ to be used for other purposes. That is, the freshness identifier γ is used no more than once for the same purpose. Hence γ is a trusted freshness.

***b) Proposed Protocol*:** The enhanced protocol is as follows. Here the server and client are two parties. Only after developing the trust in the client, the server is going to serve the client, which is the actual thing. Before that, to build the trust, it challenges the client several times and builds the trust to the required level.

Notation for the protocol:

C	Claimant
V	Verifier
TC	Timestamp of the Claimant
TV	Timestamp of the Verifier
SPW	Sub Images set={X_1, X_2, X_3…X_n}
CU	Claimant user name
N	Nonce
S	Satisfaction symbol which takes values 0 or 1 for satisfied or not satisfied

c) Protocol:

Assumption: We assume that V and C depend on the same time server.

Registration Phase:

Here the claimant C obtains a set of sub passwords SPW. Each sub password is numbered. It is with Claimant.

a) Claimant sends its CU and IP to V through a secure channel

C→V: CU

b) V accepts (if that CU is not already taken) and sends SPW through a secure channel.

V→C: SPW through a secure channel

Login Phase:

a) When Claimant wants to get any service from the server(server acts as the verifier before it accepts to provide any service), it sends its user name CU and IP number to the server.

C→V: CU

b) This message shows that the Claimant wants to communicate or it wants to get some service. Therefore the verifier must verify the Claimant. V prepares a Challenge for C

c) This Challenge will demand the part of the image which belongs to SPW.

d) The Claimant receives the Challenge and selects the sub-image, and superimpose it with a relevant sub-image from SPW.

e) After preparing this conglomeration, it sends the number to the verifier V

f) (i) The verifier V verifies the value (ii) It verifies the time stamp T_c.
If $T_c > T_v$ and (T_c - T_v) within acceptable window span, then proceed.

This is one run. By making several runs, The verifier (the server) builds trust. For every successful run, the verifier builds trust. After getting sufficient trust, it asks the last Challenge as usual but with S as '1'. That shows that the verifier (server) has developed the trust on a user (Claimant) to a satisfactory level. If it cannot build trust after several runs, it totally rejects the client.

V. EXPLANATION

Here the protocol mainly works for the building of trust. But note that again we have to give several relaxations for the theory we discussed earlier. The client has only a finite number of sub passwords, and the combinations of these passwords are also finite. So we cannot run the protocol arbitrarily any number of times. By making the SPW large, we can make the protocol close to the computational zero-knowledge protocol. But for all practical purposes, this is acceptable. In fact, no real protocol will run for a long time arbitrarily. Also, note that we do not depend on Quadratic residuosity or integer factorization. We depend only on the arrangement of sub passwords. Perhaps this is a weak Zero-Knowledge protocol, but still, that flavour is there. If it is a traditional static password scheme, we cannot create

this flavour. Similarly, the client can build trust in the verifier also.

A. Security Proof

The server may demand any one of the sub-images in any order. The client is supposed to keep the sub-images in the exact order as demanded by the server. Every time a new set of sub-images are chosen (from this space) for authentication. This is also a simulated one time pad which is highly secure. Attributes considered for comparison

a) The necessity of the Prime Numbers

This attribute is considered in view of necessity and cost to obtain large prime numbers. When the protocol uses prime numbers, obtaining large prime numbers is a difficult task, and it is also assumed that these prime numbers are not known to anybody else. An assumption may be wrong.

b) Mutual Authentication

Both server and client authenticate each other, which is necessary to avoid masquerading.

Dependency among passwords

We are using non-static passwords, i.e. every time we are using a new password. In other words, we are simulating the one-time password idea.

Repudiation means making a promise and denying it at a later stage. Non-Repudiation means if such denial occurs, the culprit must be revealed.

c) Trust

Instead of one Challenge, if one principal makes more challenges and the other principal accepts and replies as expected, then the trust develops.

V.COMPARISONS

	Lampard scheme	Yeh - Shen-Hwang"s scheme	Eldefrawy scheme	Proposed scheme
The necessity of large Prime numbers	No	No	Yes	No
Mutual Authentication	No	No	No	Yes
Building of trust	No	No	No	Yes
Dependency among passwords	High	Low	Low	Low
Non Repudiation attack possibility	Yes	Yes	Yes	No

VI. VALIDATION

Protocol Validation: This is a method of checking whether the interactions of protocols entities are according to the protocol specifications or not and to check any deadlock or live locks are occurring are not.

Here three entities are involved, namely Client, Server and adversary. From the description of the protocol, we can understand that the client obtains its SPW through a secure channel at the registration phase. This happens only on request. Before that server is not going to interact. At the login phase, only after receiving the request from the client the server is responding and challenging.

A. Adversary: This entity is modelled such that it can listen to insecure channels, store and replay.

B. Attacks: The adversary listens to the channel for sub passwords.

Protection. To obtain actual sub passwords, this information is not sufficient.

C. Deadlock: Here, the server and client are not competing for the same resources, so the conditions like hold and wait will not occur. Therefore deadlock will not occur.

D. Livelock*:* If the server and client start simultaneously, a live lock may occur. But from the protocol analysis, we understand that they are interacting in a sequential way. Therefore Livelock will not occur.

VII. CONCLUSION

In this work, we have shown how to implement the Zero-Knowledge protocols with visual cryptography. This is a model where we can eliminate the necessity of large prime numbers.

We can extend this for more than two parties communication. We can make this more powerful to colour crypto also. As man to man communications are increasing more and more, we can use this technique extensively. In several of the cases, we encounter people who don't use English. We use an image for authentication, which makes it language independent, which is a powerful aspect for real-world applications.

REFERENCES

[1] G. Ateniese, C. Blundo, A. De Santis, and D. R. Stinson, Visual cryptography for general access structures., Information and Computation, 129(2) (1996) 86–106.

[2] C. Blundo, A. De Santis, and D. R. Stinson., On the contrast in visual cryptography schemes., Journal of Cryptology, 12(4) (1999) 261–289.

[3] M. Naor and A. Shamir., Visual cryptography., in Advances in Cryptology-EUROCRYPT '94, LNCS 950, Springer-Verlag, (1995) 1–12.

[4] W. G. Tzeng and C. M. Hu., A new approach for visual cryptography., Designs, Codes and Cryptography, 27(2002) 207–227.

[5] E. R. Verheul and H. C. A. van Tilborg., Constructions and Properties of k out of n Visual Secret Sharing Schemes., Designs, Codes and Cryptography, 11(1997) 179–196.

[6] Chetanatii Hegde, Manu S, P Deepa Shenoy, Venugopal K R, L M Patnaik Secure Authentication using Image Processing and Visual Cryptography for Banking Applications 978-1-4244-2963-9/13/$26 © 2013 IEEE.

[7] C. M. Hu and W. G. Tzeng., Cheating Prevention in Visual Cryptography IEEE Transaction on Image Processing, 16(1) (2007) 36-45.