Feige-Fiat-Shamir ZKP Scheme Revisited

JOSEPH M. KIZZA*
Department of Computer Science and Engineering
The University of Tennessee, Chattanooga

Abstract
In networks and entity groupings that have sensitive resources, user identification is a crucial requirement for secure access, communication and transactions involving those resources. However, there are networks and entity groupings that require entity authentication while preserving the privacy of the entity being authenticated. There are several zero-knowledge protocols (ZKP) including the Fiege-Fiat-Shamir that authenticate an entity anonymously. We present a revised Feige-Fiat-Shamir ZKP scheme for the Airborne Networks (ANs) that reduces the ping-pong effect in the scheme and speeds up the growth of the Verifier trust of the Prover, thus making the authentication process faster and more efficient.
Key Words: Mission critical, time, authenticity, anonymity, zero knowledge, authentication.

IJCIR Reference Format:
Kizza, Joseph. M. Fiege-Fiat-Shamir Revisited. Journal of Computing and ICT Research, Vol. 4, No. 1, pp. 9-19. http://www.ijcir.org/volume4-number1/article2.pdf.

1. INTRODUCTION

In agile networks and other entity groupings that are mission driven, time sensitive, ad-hoc and self-organizing like in the Airborne Networks (ANs), authenticity, anonymity and accountability are essential and crucial and probably more so than in other similar networks that are less mission critical and time sensitive. In these kinds of networks, standard cryptographic authentication protocols like PKI cannot work effectively.

An Airborne Network is a self-forming, self-organizing, and self-generating, mission-critical and time sensitive network of airborne entities as nodes joining and leaving the network as they enter and exit specific regions. The network consists of dedicated tactical links, wideband air-to-air links, and ad-hoc networks constructed by the Joint Tactical Radio System (JTRS) networking services (Wikepedia [2009]). In addition, the JTRS is a software-defined radio that normally works with many existing military and civilian radios with the help of an integrated encryption and Wideband Networking Software that also provides system performance analysis and fault diagnostics automatically, reducing the demand for human intervention and network maintenance (Austin Mohr, [2007]).

There several efficient proofs including classical formulations of NP, interactive proof systems, computationally-sound proof systems, and probabilistic checkable proofs that can be good candidates for the ANs. However, the most suitable are proofs in the interactive proof systems. In particular, the zero-knowledge proof system (ZKP defined as Odel Goldreich [1993]: *For a language L a zero-knowledge proof system is a pair (P, V) of interactive machines, so that V is probabilistic polynomial-time, satisfying*

- Completeness:*For every x Є L the verifier V always accepts after interacting with the prove r P on common input x .*

* Author's Address: Joseph M. Kizza, Department of Computer Science and Engineering, The University of Tennessee-Chattanooga, Chattanooga, TN 37403, USA,*Joseph-kizza@utc.edu*.

International Journal of Computing and ICT Research, ISSN 1818-1139 (Print), ISSN 1996-1065 (Online), Vol.4, No.1 pp. 9-19, June 2010.

• Soundness:*For every x not in L and every potential prover P*, the verifier V rejects with probability*
at least ½ after interacting with P on common input x .*

The *Prover* P is a machine that is seeking authentication that must be done by the Verifier V upon presentation of evidence *by* the *Prover* to the Verifier that does not give away the identity of the *Prover*. Based on the evidence provided by the *Prover*, the verifier may get satisfied and authenticate the *Prover* without knowing the identity of the *Prover*. Unlike the Public Key Infrastructure (PKI), a framework for creating a secure method for exchanging information based on public key cryptography. The key role for the PKI is the certificate authority (CA), which issues digital certificates that authenticate the identity of entities over a public communication system as the Internet. ZKP does not involve this CA acting as a third party.

In ZKP, the *Prover* presents to the *Verifier* a publically generated and known part of the token. Upon receiving the public part of the *Prover*'s token, the Verifier then may require the *Prover* to produce a *response*, to a Verifier's randomly generated *challenge* based on the *Prover*'s public part of the token. On receiving the *Prover*'s response, the *Verifier* then validates the response using no more than the publicly known information. Two possible outcomes are either "yes" or "no". If a "yes" answer is produced, the authentication process is successfully over and the token is now *spent,* meaning, it has served its purpose and it is done. Any other authentication needed by this entity requires a new token to be generated. If the outcome is a "no" a new round of authentication may be required. This may go on for a number of back and forth ping-point exchanges until success. A successful ZKP authentication leads to the *Prover* remaining anonymous to the verifier.

Several scholars (Park Choonsik [1992], Dev Anshul and Suman Ray [2005], M. J. Fisher [2005], Hannu A. Aronnsson [2000]) have written about the problems of ZKP protocols including their inefficiency in exchanging tokens during the *challenge-response* sessions. Such problems tend to get worse as the network sizes grow.

In this paper we propose a new approach to the Feige-Fiat-Shamir ZKP scheme that aims to reduce the number of *challenge-response* ping-pong exchanges and thus create a speed-up in the building of trust and increasing the effectiveness of the scheme.

The rest of the paper is organized as follows. We will introduce the Feige-Fiat-Shamir ZKP Scheme in Section 2. Then we will explain the new improvements to the Feige-Fiat-Shamir Protocol scheme in Section 3. Security and trust analysis will be given in Section 4. Problems with Parallel Execution of Zero-Knowledge Proofs are discussed in Section 5. Finally we conclude and discuss possible extension in Section 6. References are in Section 7.

2. RELATED WORK

Several interesting zero knowledge proofs are of note here starting with Hannu Aronsson,s (Wikipedia [2009]) work which explains zero knowledge proofs starting with the basics including a summary of all the major zero knowledge studies. Li Lu et al [2006] discuss zero knowledge authentication in P2P systems. However, they use a modified ZKPI scheme that uses a key exchange but with no third party. Austin Mohr [2007] gives an in depth survey of zero knowledge proofs ranging from Graph Isomorphism, Graph 3-colorability, All Languages in NP and Fiat-Shamir protocol. None of these works attempts to extend the speed of authentication as ours attempts to do.

3. FEIGE-FIAT-SHAMIR PROTOCOL

The Feige-Fiat-Shamir Identification Scheme is a classical, practical and widely used modular arithmetic ZKP scheme developed by Uriel Feige, Amos Fiat and Adi Shamir in 1988. Following the principle of all zero-knowledge proofs, the Feige-Fiat-Shamir Identification Scheme involves two entities, the *Prover*, and the *Verifier*. The *Prove*r possesses a secret token and is seeking authentication and must prove to another entity, the *Verifier*, who must authenticate the *Prover* based upon the secret token the *Prover* has, through a series of challenges without getting to know the P*rover*'s secret token.
Procedure

The Feige-Fiat-Shamir process involves the *Prover* choosing two large Blum prime integers *p* and *q* where each is of the type 4g+3 and *gcd*(g, 3) =1. Also the *Prover* chooses two security integers *k* and *t*. Then computing the product *n* = *pq*. The protocol then uses this n in the modular arithmetic that follows.

The congruence relation

Two integers *a* and*b* are said to be *congruentmodulon*, if their difference *a* − *b* is an integer multiple of *n*. That is both numbers have the same remainder when divided by n. This is then expressed as:

$$a \equiv b \pmod{n}.$$

A good example are integers a = 38 and b = 14. Then 38 ≡ 14 (mod 12) shows that (a, b) form a congruence class.

Using this *n*, choose or create secret vector $s = \{s_1, \cdots, s_k\}$ with $gcd(s_i,n) = 1$, and c = { $c_{i=1,\,k}$ | c_i Є (0, 1)}. *n* is then made public. Now compute the vector *v* = { $v_1, v_2, \ldots v_k$ | $v_i \equiv s_i^2 \pmod{n}$} where ≡ is a congruency relation between v_i and s_i^2, meaning that v_i and s_i^2 have the same remainder upon division by n.

The vector *v* then is sent to the *Verifier*. The difficulty the *Verifier* may encounter in recovering the vector *s* as it involves the computation of the modular square root without knowing the modulus' factorization. The Feige-Fiat-Shamir procedure goes as follows [7].

1. *Prover* chooses a random integer *r*, a random sign b Є {-1, 1}and computes $x \equiv (-1)^{ci,\, i=1,\, k} \cdot r^2 \pmod{n}$. *Prover* sends this number to Verifier.
2. Verifier chooses numbers $a_1, \cdots, a_k$ where a_i equals 0 or 1. *Verifier* sends these numbers to *Prover.*
3. *Prover* computes $y \equiv r s_1^{a_1} s_2^{a_2} \cdots s_k^{a_k} \pmod{n}$. *Prover* sends this number to *Verifier*.
4. *Verifier* checks that $y^2 \equiv \pm x v_1^{a_1} v_2^{a_2} \cdots v_k^{a_k} \pmod{n}.$

This procedure is repeated with different *r* and a_i values until *Verifier* is satisfied that *Prover* does indeed possess the modular square roots (s_i) of his v_i numbers.

4. THE REVISED FEIGE-FIAT-SHAMIR PROTOCOL

The Ping-Pong Problem

While the Feige-Fiat-Shamir Identification Scheme is the most celebrated, classical, practical and widely used modular arithmetic ZKP scheme, it suffers from the *ping-pong problem.* The Ping Pong problem in the ZKP solutions is caused by repeated, sometimes uncontrolled, challenge-response exchanges between the *Verifier* and the *Prover*, as the *Verifier* tries to get as much information as possible from the *Prover* in order to complete the authentication process in the shortest time possible but in the most sure way and as the *Prover* tries to provide the needed information for the authentication process without revealing the *Prover*'s identity. The ping-pong problem is a resource guzzler. Given that the AN environment is time critical, the authentication process needs to be comprehensive, precise and take a short time. But in the Feige-Fiat-Shamir this cannot be accomplished because, there is no sure way of making the process short. In some cases, the authentication process may be short but not always. We propose a procedure that speeds up this process and it builds the trust quickly. The scheme still involves two entities, the *Prover*, who possesses a secret token and is seeking authentication and must prove to another entity, the *Verifier*, who must authenticate the *Prover* based upon the secret token the *Prover* has, through a limited series of challenges without getting to know the *Prover*'s secret token.

Procedure

The revised Feige-Fiat-Shamir protocol still involves the *Prover* choosing two large Blum prime integers *p* and *q* where each is of the type 4g+3 and *gcd*(g, 3) = 1. Also the *Prover* chooses two security integers *k* and *t*. The *Prover* then computes *n* = *pq*. Using this *n*, the *Prover* then chooses or creates a secret vector $s = \{s_1, \cdots, s_k\}$ with $\gcd(s_i,n) = 1$. *n* is then made public. Further the *Prover* still computes the vector *v* = {$v_1, v_2 \ldots vk$ | $v_i \equiv s_i^2 \pmod{n}$}. The vector *v* is then sent to the *Verifier*. Like in the original Feige-Fiat-Shamir protocol, the Holy Grail of the protocol is the difficulty the

Verifier encounters in recovering the vector *s* as this involves the computation of the modular square root without knowing the modulus' factorization. The revised Feige-Fiat-Shamir protocol then goes as follows.

1 *Prover* chooses a random integer *r*, a random sign *b* in set {-1, 1}and computes $x \equiv b.\ r^2 \pmod{n}$. *Prover* sends this number to *Verifier*.

2. Each time the *Verifier* chooses a challenge, there are ${}^{k}C_{i=1,\,k}$ possible ways to choose the vector $a = \{ a_1, \cdots, a_k \}$ where a_i equals 0 or 1. These choices are represented by the 0-1 matrix M = $\{(a_{ij}) = (0,1)|\ (i, j) = 1, k\}$. From these rows of M, the *Verifier* randomly selects a number of rows forming a sub-matrix D = $\{(a_{ij}) = (0, 1)\ |i=1, k, j=1, f.\}$ of M to concurrently send to the *Prover* as the challenge.
3. Instead of the *Prover* computing only one number y, the *Prover* uses the received matrix D to compute a vector
   $y=\{ y_1,\ y_2,\ \ldots\ y_f \mid y_i \equiv r \pi s_i^{a_{ij}} \pmod{n}, i=1..k \text{ and } j=1, f\}$.
   The *Prover* then sends this vector *y* to the *Verifier*.
4. *Verifier* checks that each of the vector components of *y* satisfies $y_i^2 \equiv \pm x \pi v_i^{a_{ij}} \pmod{n}, i=1..k$ and j=1, f}.

This procedure may be repeated with different *r* and different sub-matrices D = $\{(a_{ij}|\ i=1, k, j=1, f.\}$ of M until the *Verifier* is satisfied that the *Prover* does indeed possess the modular square roots (s_i) of his v_i numbers.

An Example:

1. Suppose either the *Prover* or a trusted center T selects the primes p = 139, q = 347, and publishes n = pq = 48233.
2. The *Prover* chooses Integers k = 3 and t = 1 as the needed security parameters.
   Then the *Prover* does the following:
   - Selects 3 random integers s_1=87, s_2= 21649, s_3 = 523, and 3 bits b_1 = 1, b_2 = 0, b_3 = 1.
   - Computes v_1 = 7569, v_2 = 10310, and v_3 = 32364.
   - The *Prover*'s public key then is (7569, 10310, 32364, 48233) and the *Prover*'s private key is (87, 21649, 523).
3. The *Prover* chooses integers r = 3209, b = 1, and computes x = 24052, and sends this to Verifier.
4. Since k was chosen as 3, the *Verifie*r has the following possible matrix
   M = {0 0 0, 0 0 1, 0 1 0, 0 1 1, 1 0 0, 1 0 1, 1 1 0, 1 1 1 }
   of choices of challenges.
5. Suppose the *Verifier* chooses a sub-matrix
   D= { 0 1 0, 0 1 1, 1 1 1}
   and sends it to the *Prover*.
6. The *Prover* must compute a vector y = { y_1 , y_2 , y_3 } = {16121, 5788, 36823}
7. The *Verifier* computes (y_1^2 , y_2^2 , y_3^2) = (259886641, 33500944, 1355933329). Also computes ((24052*[10310], 24052*[10310*32364], 24052*[7569*10310*32364])
8. To accept the *Prover*'s identity the following pairs of numbers must be in the same congruency classes mod 48233:
   a. (259886641 and 24052*10310)
   b. (3350094 and 24052*10310*32364)
   c. (1355933329 and 24052*7569*10310*32364)

5. SECURITY AND TRUST ANALYSIS

Using the scheme we have outlined above one achieves tremendous savings first in time speed up and rapid build up of the trust and also in the security which is not compromised. Based on the Feige-Fiat-Shamir protocol on which this scheme is built, the *Prover* does not give any useful information to the *Verifier* during the whole process. If there is any interception of the communication, the interceptor can learn no more information than what the P*rover* gives out. The knowledge of the *Prover*'s secret cannot leak because it is not communicated to the *Verifier*. The new scheme adds nothing that would change this fact. Therefore the security of the scheme is assured.

The major contribution of this scheme is a quick build up of the *Verifier* trust and confidence for the *Prover*. We can calculate this trust build up as follows. Suppose the security integers chosen by the *Prover* are k and 1. This means that the Verifier has k! ways of choosing a 0-1 vector of length k. This is what we referred to in the protocol as matrix M. Suppose further that the Verifier decided to send several of these vectors at a go. The *Verifier* can decide to send to the *Prover* a sub-matrix D of M consisting of these choices, the *Prover* then sends back the vector $y = \{y_1, y_2 \ldots y_f\}$. At each parallel burst resulting in matrix D of f rows, with k bits each, which the Prover returns correct, the *Verifier*'s trust/confidentiality grows as $\{1 - ((1/2)^f)^k\}$. So for t bursts of parallelism, the contribution to trust increases by $\{1- (((1/2^f)^k)^t\}$. This is a good speed up which makes the authentication process take less time.

6. PROBLEMS WITH PARALLEL EXECUTION OF ZERO-KNOWLEDGE PROOFS

Parallel executions of zero-knowledge in Feige-Fiat-Shamir protocol have been proposed and they are certainly attractive and they reduce the number of ping pong messages between *Prover* and *Verifier* to only 1 1/2. However, it has also been proven that they may not be fully zero knowledge (Joe Kilian, Eriz Petrank and Ransom Richardson [2001], M. Konidala Divyan [2003]). However, there have also been concurrent zero knowledge techniques that preserve zero knowledge (Li Lu, Jinsong Han, Yunhao Liu, Lei hu, Jinpeng Huai Lionel Ni and Jian Ma [2006]). Our approach consists of bursts of parallelism while maintaining serial execution of the Feige-Fiat-Shamir, hence maintaining zero knowledge. Note that the difference between the serial and full parallel versions of the protocol is that in the latter, the *Verifier* gets to know all of the x_i's before choosing the b_i's. This does not happen in our approach. The Verifier chooses matrix D with no knowledge of all the x_i's. We generate bursts of parallelism consisting of matrices D with no more that $f < k$. The matrices D can be calculated several times depending on the number of bursts of parallelism that can occur during an entity authentication. This number, however, cannot be more that t.

7. SIMULATION AND RESULTS

The full simulation to support this paper was planned to be carried out in two phases as follows:

(1) In phase one we planned to generate and test the C code developed by Daniele Raffo [2002] at LIX, Ecole Polytechnique, France. This code was modified and run in two modes, the single processor mode and the multi-processor distributed environment. In the single processor mode, we made several runs of the modified code and in the distributed environment, we developed the handshake protocols for the exchange of information between the Prover and the Verifier processors in a client-server model.

(2) In phase two, we plan to develop new C++ code for the revised Feige-Fiat-Shamir zero-knowledge protocol discussed in this paper. Next we will develop the handshake protocol for the new FFS. We ill then develop the test databed. With the test bed we will do:
- Distributed testing
- Speed up protocol development
- Testing using UAVs as needed and where possible.
- Analysis of results

In phase one, we have ran the modified C code on a trial basis and Figure 3 shows the outputs of those runs for T =10 and K =10. The results in Figure 3 show the full range of outputs which included modulus n, both the computed public and private keys after each of the ten iterations, the challenge from the Verifier, the response from the Prover, the verification by the Verifier and the acceptance by the Verifier. Since the growth of the output data was astronomical, we decided not to continue with this kind of experimentation.

Figure 3 (below): Row output from the Modified C Code.

Publishing modulus:

17976929205606008571648315122074122220102965845447940859554850840430813475670227100983
50184399271343467327287010688126007525809674107012108387332202665407984759216729068028
82425893135819150850301344123141412498892491036325633388257822297629447413753876149348
54339115723519240541066133028625182964279986761052 9

Computing keys

Public key:
1325625555726427167602409069216477383076579971304311722752863292651094976110223310819670413057761792102446546320619998748411283056671496121877666821840103407959409519981908786112124403633566902286466765434780727350808861805796186111853160858368149106687007721785965602349251552438499092925278075507868773255981363040399630229814159786352017780911194761028820357101208107965118121874688440252376220058072988193501083406566024325204606396729189849440291576507536446421239343452602583277613058987846808791306560980147577320147848477920446984605426214323428801939239897129197280072580967774905613361071379688676362839020489152462554870962359841888139328838089678339265219026503282406363651763056218420844130375214315104013861149166043737298368977761673750514279370537015098821581567292055666783817889373551332606755700246724154724186352287786015971103068270800980688752396069141086920516938867505853165773780902278687771285947420410852062878736044524314651638729447161297468389151454417802470349123506768218846068339110905692312149821768463815469631355320209741130128167547179066954441117053863271572874022042238631338281993714994883613132412848608845148483213000863465475842195700373682942121828006388464532269902281817588111155562062287017394490629576510437597664352437135834914159329103498549131612574992273407611080476852400701908655688254834182964030122592281052993772384108908446998328365675192775218551834820692330419502787327177873516203547060541946789181886640814250825382029365102814962504792642549942146927731917465847164750212592853251011192529250098233222560673627420816678837136040240815033214924870733065091217240716506008360110042854141635871527896034462547936586908792446881628437929206369244138753559419775373809059390754551629755460753520862626461725939972056642314720257811758574814490238747930102237552917368306150608252623220437118539792822503344974732701231405311799192202978414254399253894641868595667210911014724422953218788561864433990710380585781658514535605173851050196830591298915521614177714244986963187242555290449794839136506431869607286660820856276298591254454572626502764417411653242477282844391921437713347501056915587463871075507769317309492597441031008003085647525428435555477051077667163982730141083956675053169152468498678481836164149403102781116890658043996620446432123302772274904580602048853758373507470784695886513764158608352222181732666220688793988274224773568634990989151282593567730936015328815364921718026790141098972893596017109148207572442309009686117360022096200899570941288416101391003975383768152279674908896757871410956114604643490469758787331966014627586019381169445543698267437933139135290113273593673984042001004077126652252813015191124082891286652846653672371554644166400993757337504595597246022460444794694869785121025584545166566764469235060056186772671705965645818045757226722964643433543220396128347228784259841133878308587422191553209646448919081033341443829160398028032957594605522027019806460974471618666362227087465044490552882058611657174179550983513384320904827797158689537747787992311139996551112922389758085375239245241Private key:
1509065172426750530997184852818281763976288348406372050434621087205456771253588726176489265163983867716116255306926766719477521261840399065391330144079770376099761751215442826877981738363787063115066658886546396061661374728131024652132173345754444986115944436011033177811066369111398358148531511039612189391331212946952084800353848686956889576281594996014169627644599876094753114255485716941221893136564775349795845935450665813086178063436346366065512233056736461659419323720660335768596576290810297389822595126022597737722063931686360491382793095922723028253192119896767408290847272964868484796815942952909768991176838156192001585467784579845106959721530679367158352771193914226236612832021647838464083791633674782847427932435078326922010868921983987891881307204399046603073739967290637606503164678179518816642292522021767149970747002768545690951029354992973819220887304803952261707336114172928800409692135487768743212955435166

2626163218025003952764808051705139080645925646378962302441086594713788430265639254094943002276382123383447276738325340739212891357292380075992719396340716469912291393735874963530248072511739693392434965120311493590041923118777936294041556862101854256581658130215611129323481511731893603129607028941246538144811152088734908610854001180587543354757635655365251931096578533904100419123149331085750161689156502327143652757848522471652741143560384839660254931268353605866312961795144021714457926249442502604839711334714571200186942729539001025590285925064025947580170227054091131770597885859797007682173492958280738924561043054098021770135088174771098105833891719472210682001998571405321296160087742147272495420840963028844919052949860446151139200676954004460783152564727006693574998340538437252767441661098866046034760861558682955053462937500835058349521398944892675431837895176036941162551971473550080587989115712433648396977555126880482692408528561987590492057123648212315771041303216190981542417081950858276741638083293798742876989903323615055621272177672248330084193447448864311582285832059864445869276458231405997136394987211967912986614047232992950838418542896590241758204040896430209721228979078693552692153469664355418355201312421129368255091043051453554660497780252168560899872636040211651543283814040902149310892082884243764607656750615442261907060756047417527764319333015713860268566567877152126610546852339217807288022127853424796948792724336507164222694916480946104388172674846491531096561148659130946473477303795819122557420978423911219504989159304145804447904961229020560947647075917794745157481575664457355960220487813240338259576653921987724440975264863705717286172208157103413695196039316076755204285653374196275646365405746176477367421986717999839672254561657083513262383331523251026116864694639635376416585562606028125018079094674525966457423698969402991813166828200639023406606582801940676118130300251139098434855327309395359505992903105215139157468031852231707045936130256625673325627583966473350875992821807689689402501086769056633388278648578959356081199014886090277554002831330022231419900749205277003448738458216286098718567909392354774Witness :1008944640098410497152169431152416751194146033174694516945601792058144640132780917392745476319978048224998062463459693576765658363118819438361461271262703463212570763281231301123673021181184245731831188059021236323470506429895320987271717948100638592136300488493235330684475221977219513959402914474026606857474Challenge : 1001001110Response :7810173275356886115905966317186787418242627066881777507829842487368944771924046568396932489913456343487198548542240535084373605548162504832332491553866493744768880279715678781300927802226020260659284716471898158165417801569824660704643132625629316340689690919360148727479600709380214702228401577777633888289711Verification:1008944640098410497152169431152416751194146033174694516945601792058144640132780917392745476319978048224998062463459693576765658363118819438361461271262703463212570763281231301123673021181184245731831188059021236323470506429895320987271717948100638592136300488493235330684475221977219513959402914474026606857474Authentication successful!

Next we restructured and expanded this code to convert it into C++ so that we could run both the Prover and the Verifier codes as threads. Several runs were made with varying sizes of K, T and time tamps. Figure 4 shows selected outputs of these runs except the varying time stamps for each change in T and K. We are showing a selected number of runs of varying sizes of Ts and Ks using a single core thread environment:

Figure 4 (below): Selected Runs with Varying Sizes of Ts and Ks Using Threads

T = 10, 100, 500, 1000, 2000, 5000.

K = 10, 20, 50, 70, 100, 500

Sample Runs:

1. T=5 and K = 4

Feige-Fiat-Shamir ZKP implementation
Iteration 0: 06.12.2009 22:24:27
Authentication successful.
Iteration 0 finished at: 06.12.2009 22:24:27

Iteration 1: 06.12.2009 22:24:27
Authentication successful.
Iteration 1 finished at: 06.12.2009 22:24:28

Iteration 2: 06.12.2009 22:24:28
Authentication successful.
Iteration 2 finished at: 06.12.2009 22:24:28

Iteration 3: 06.12.2009 22:24:28
Authentication successful.
Iteration 3 finished at: 06.12.2009 22:24:28

2. T=10 & K = 8
Feige-Fiat-Shamir ZKP implementation
Iteration 0: 06.12.2009 23:31:07
Authentication successful.
Iteration 0 finished at: 06.12.2009 23:31:07

Iteration 1: 06.12.2009 23:31:07
Authentication successful.
Iteration 1 finished at: 06.12.2009 23:31:07

Iteration 2: 06.12.2009 23:31:07
Authentication successful.
Iteration 2 finished at: 06.12.2009 23:31:07

Iteration 3: 06.12.2009 23:31:07
Authentication successful.
Iteration 3 finished at: 06.12.2009 23:31:07

Iteration 4: 06.12.2009 23:31:07
Authentication successful.
Iteration 4 finished at: 06.12.2009 23:31:07

Iteration 5: 06.12.2009 23:31:07
Authentication successful.
Iteration 5 finished at: 06.12.2009 23:31:07

Iteration 6: 06.12.2009 23:31:07
Authentication successful.
Iteration 6 finished at: 06.12.2009 23:31:07

Iteration 7: 06.12.2009 23:31:07
Authentication successful.
Iteration 7 finished at: 06.12.2009 23:31:07

K	Seconds
10	1
50	3
100	3
500	10
1000	34
2000	119
3000	296
4000	509
5000	807

Table 1: Changing values of K, T =20 and the time stamps.

To learn more of these changing timestamps, we extended this algorithm so that it can work in the multi thread and truly distributed environment of a client-server model by developing communication protocols and handshake necessary in a networked environment. Under this environment, the Verifier runs as the server and the Prover as the client. In this Client-Server model, we tested the algorithm on differing sizes of both K and T. As we did this, the time stamp was taken at the start and end of each exchange between the client and the server until the Verifier accepted the Prover.
In Table 1 and corresponding Figure 5 we show results of runs in this distributed client-server environment with varying sizes of K.

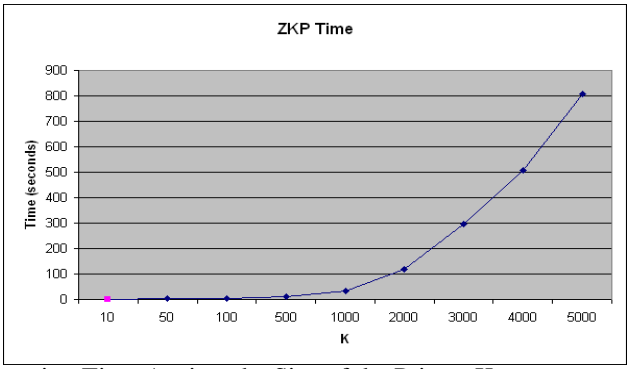

Figure 5: Growth of Computation Time Against the Size of the Primes K.
From Figure 5 above, there is compelling evidence that the larger the size of the integer strings, the longer it takes for the authentication of the entity.

9. ZKP COMPARED TO PKI

We have not compared ZPK with suitable PKI yet but this will be our next step.

10. FUTURE WORKS

As we pointed out earlier, this work has so far dealt with the original Feige-Fiat-Shamir. The simulation of the Feige-Fiat-Shamir discussed in full is to be tackled in the planned phase II of this work. This will be the starting point for our future work. In particular, we will generate new code C++ code for the revised Feige-Fiat-Shamir zero-knowledge protocol. Next we will develop the handshake protocol for the new FFS. We ill next develop the test databed. With the test bed we will do:

- Distributed testing
- Speed up protocol development
- Testing using UAVs as needed and where possible.
- Analysis of results

- Major testing

The purpose of this work was to revisit and revise Feige-Fiat-Shamir's original ZKP scheme in order to remove the ping-pong effect and make it faster to use. To meet this goal, after we get results in phase II, we will compare the duration of authentications in the distributed client-server model for the authentication of the Prover by the Verifier in both phase I and II. The differences we get in this comparison will demonstrate the time savings when our revised Fiege-Fiat-Shamir algorithm is used.

Zero Knowledge Protocols are designed to work between two parties, the Prover and the Verifier. Through verification rounds the Prover attempts to convince the Verifier he possesses a secret. Over time the Verifier may trust the Prover has a secret and allow the Prover to communication with the Verifier.
The trust the Verifier has for the Prover does not transfer to other nodes of the network. Further work, as a continuation of this project, is needed to look into how the Prover earns the trust of the entire network once accepted by the Verifier.

11. CONCLUSION

In this paper, we have produced a revised Fiege-Fiat-Shamir protocol scheme in which the Verifier instead of choosing one 0-1 vector as a challenge to the *Prover* now chooses a sub-matrix whose rows are individual challenge vectors. Increasing the number of challenge vectors sent to the Prover at once speeds up the growth of the Verifier's trust of the *Prover* making the whole authentication process a lot faster and more efficient. We have also developed test programs to run the algorithm in a multi thread environment. Further we have developed the necessary protocols to run the algorithm in a client-server model which gives it a real distributed environment. As we did this we took timestamps of each run and computed time growth computation with the increasing size of the primes used. Looking at the times graph raises more and interesting questions that require extending this study. Such improvements may include finding more efficient and economical ways to move matrix D from the *Verifier* to the *Prover*. In our next attempt, we will focus on ways to move this matrix more economically. We want to investigate ways to either decompose the matrix or find some other lossless compression that will cut down on the amount of data passed.
There are several other issues that are also attracting our interest. Some of these include:

- Since ANs are P2P networks, does authentication of an entity by another network entity lead to global network authentication of that entity? If not, how do we handle subsequent authentication of that entity?
- If an entity leaves the network constellation and comes back in a very short time, does this mean a new authentication? Do we need to generate a session identification code to reduce on the number of authentication requests?

12. REFERENCES

AIRBORNE NETWORKING.http://en.wikipedia.org/wiki/Airborne_Networking

AUSTIN MOHR. " A Survey of Zero-Knowledge Proofs with Applications to Cryptography". http://austinmohr.com/work/files/zkp.pdf

CHOONSIK, PARK. A *Roubust Identification Protocol Without a Highly Reliable Trusted Center*. Singapore ICCS/ISITA 1992, IEEE, 1992.

DANIELE RAFFO, 2002: Master of Science in Computer Science, specialization in Networking - with honors. "Digital Certificates and the Feige-Fiat-Shamir zero-knowledge protocol".Université Paris-Est Marne-la-Vallée.

DEV ANSHUL AND SUMAN RAY.*A ZKP-based Identification Scheme for Base Nodes in Wireless Sensor Networks*. ACM SAC '05, March 13-17, 2005.

FISHER, M. J.*Zero Knowledge Interactive Proofs,* http://zoo.cs.yale.edu/classes/cs467/2005s/course/lectures/ln_week10.pdf

HANNU A. ARONSSON. "Zero Knowledge Protocols and Small Systems".http://www.tml.tkk.fi/Opinnot/Tik-110.501/1995/zeroknowledge.html *http://www.iacr.org/archive/eurocrypt2000/1807/18070424-new.pdf*

IVAN DAMGºARD.*Efficient Concurrent Zero-Knowledge in the Auxiliary String Model*

JOE KILIAN, EREZ PETRANK AND RANSOM RICHARDSON.*On Concurrent and Resettable Zero-KnowledgeProofs for NP.,*http://arxiv.org/abs/cs.CR/0107004

KONIDALA M. DIVYAN .*ComparativeStudy on Zero-KnowledgeIdentificationProtocols* http://caislab.icu.ac.kr/Lecture/data/2003/spring/ice514/project/f02_divyan.ppt.

Li Lu, Jinsong Han, Yunhao Liu, Lei hu, Jinpeng Huai Lionel Ni and Jian Ma. "Pseudo Trust: Zero-knowledge Authentication in Anonymous P2Ps". *IEEE Transactions on Parallel and Distributed Systems.* Vol. 19, No. 10, October 2006.

Oded Goldreich.*A Taxonomy of Proof Systems (part 1).* http://delivery.acm.org/10.1145/170000/165000/p2-hemaspaandra.pdf?key1=165000&key2=0808024421&coll=GUIDE&dl=GUIDE&CFID=39183453&CFTOKEN=53850464

RETRIEVED FROM "http://en.wikipedia.org/wiki/Feige-Fiat-Shamir_Identification_Scheme".

SHAFI GOLDWASSER AND YAEL TAUMAN KALAI.*On the (In)security of the Fiat-Shamir Paradigm*. Proceedings of the 44th Annual IEEE Symposium on Foundations of Computer Science (FOCS'93).

WADE TRAPPE, LAWRENCE C. WASHINGTON, *Introduction to Cryptography with Coding Theory* (Prentice-Hall, Inc., 2003), pp. 231–233.