The ITB Journal

Volume 3 | Issue 1 Article 3

2002

Identification Protocols in Cryptography

Michael O'Donnell

Identification Protocols in Cryptography

Michael O'Donnell

School of Informatics and Engineering, ITB Blanchardstown, Dublin, Ireland

Abstract

In this paper we examine the role of Identification Protocols in the field of Cryptography. Firstly, the rationale behind the need for Identification Protocols is discussed. Secondly, we examine, in detail, challenge-response protocols, based upon zero-knowledge proofs, that form a subset of Identification Protocols in general. Thirdly, the mathematical tools necessary for the understanding of how these protocols work is given. Finally, we discuss four main Identification Protocols: Fiat-Shamir, Feige-Fiat-Shamir, Schnorr and Guillou-Quisquater. This discussion includes the theory, practical examples and the security aspects of each protocol.

1. Introduction

"If you think cryptography is the answer to your problem, then you don't know what your problem is."

Peter G. Neumann
Quoted in the *New York Times*

[12]

At the moment there are a plethora of terms, definitions and even some disagreement about what constitutes message authentication, data origin authentication, transaction authentication, key authentication and entity authentication or identification.

For consistency and clarity, we will use the definitions as outlined in the "Handbook of Applied Cryptography" by Menezes *et al*. [1]

2. Authentication Methods

Authentication is any process by which an entity (which could be a person or indeed another computer) establishes its identity to another entity. Authentication can generally be divided into two main areas:

1. Entity authentication or identification
2. Data-origin authentication

Authentication in the broadest sense encompasses not only these two areas, but also protection from all **active attacks**. Active attacks involve some modification of the communication between source and destination or indeed the substitution of an authentic communication by another communication entirely. This contrasts with encryption, which only provides protection from **passive attacks**. Passive attacks involve eavesdropping or monitoring of a communication between source and destination. [2]

2.1 Data origin authentication (message authentication)

Data origin authentication is a type of authentication whereby a party is corroborated as being the original source of specified data created at some (typically unspecified) time in the past. By definition, data origin authentication includes **data integrity** - which guarantees that data has not been altered in an unauthorised manner since the time it was created, transmitted or stored by an authorised source.

Message authentication is a term that is similar to data origin authentication. It provides data origin authentication and data integrity but does **not** provide **uniqueness** or **timeliness**.

Uniqueness guarantees that the source of the data can be identified with an established degree of confidence and that the source we are dealing with is who they purport to be. Timeliness guarantees that the data was sent by the source at a time which can be verified by both the source and destination.

Two common methods of providing message authentication include:

1. Message authentication codes (MACs)
2. Digital signature schemes

While MACs and digital signature schemes may be used to establish that a specified party generated data at some time in the past, they establish no uniqueness or timeliness guarantees. These methods alone cannot detect for example **message replay** (where a message is passively captured and is subsequently retransmitted to produce an unauthorised effect).

2.2 Transaction authentication

Transaction authentication not only provides message authentication but also uniqueness and timeliness of data, thus preventing undetectable message replay. The uniqueness and timeliness guarantees are provided by random numbers in **challenge-response protocols**, sequence numbers and timestamps. [1]

A typical scenario, involving the use of challenge-response protocols in a private-key cryptosystem such as DES (Data Encryption Standard), consists of the following three steps: [3]

1. Bob chooses a challenge, *x*, which is a random 64-bit string. Bob sends *x* to Alice.

2. Alice computes
$$y = E_k(x)$$
and sends it to Bob.

3. Bob computes
$$y' = E_k(x)$$
and verifies that
$$y' = y$$

2.3 Entity authentication (identification)

Entity authentication is the process whereby one party is assured of the identity of a second party involved in a protocol and that the second party has actually participated in the execution of that protocol. These protocols, which are more commonly known as **identification protocols,** use **asymmetric** techniques, but do not rely on digital signatures or public-key encryption. Neither do they use sequence numbers or timestamps. Instead they use random numbers, both as a challenge and a commitment, based upon interactive proof systems and zero-knowledge proofs. [1]. Challenge-response protocols based upon zero-knowledge proofs are the main topic of investigation in this paper.

2.4 Key authentication

Key authentication is the property whereby one party is assured that no other party aside from a specifically identified second party (and possibly additional identified trusted parties) may gain access to a particular secret key. This requires both the secrecy of the key and identification of those parties with access to it. The identification requirement here differs in one very important respect from that of entity authentication outlined above. Here the requirement is knowledge of the identity of parties that may gain access to the key, rather than corroboration that actual communication has been established with such parties.

Kerberos is a key transport protocol that is based on **symmetric** techniques. In symmetric techniques, the same key is shared between the two parties, Alice and Bob, with the proviso that the key is kept secret from an adversary such as Oscar. It may, however, be shared with a trusted third party or a key distribution centre (KDC). Kerberos is an example of a challenge-response scheme providing both entity authentication and key establishment based on symmetric encryption.

Public-key techniques may also be used for challenge-response based identification providing both entity authentication and key establishment.

The table summarises the properties already defined.

	Property		
Type of authentication	**Identification of source**	**Data integrity**	**Timeliness or Uniqueness**
Message authentication	Yes	Yes	----
Transaction authentication	Yes	Yes	Yes
Entity authentication	Yes	----	Yes
Key authentication	Yes	Yes	Desirable

3. Identification Objectives

Identification is the process through which one ascertains the identity of another person or entity. In our daily lives, we identify family members, friends and coworkers by their physical properties, such as voice, face or other characteristics. These characteristics, called biometrics, can be used on computer networks with special hardware. Entities on a network may also identify one another using cryptographic methods. An **identification scheme (or**

protocol) allows Alice to identify herself to Bob in such a way that someone listening in cannot pose as Alice later. One example of an identification scheme is a **zero-knowledge (identification) protocol.** Zero-knowledge protocols allow a person (or a server, website, etc.) demonstrate they have a certain piece of information without giving it away to the person (or entity) they are convincing.

Suppose Alice knows how to solve the Rubik's cube and wants to convince Bob she can do so without giving away the solution. They could proceed as follows. Alice gives Bob a Rubik's cube which he thoroughly messes up and then gives it back to Alice. Alice turns away from Bob, solves the puzzle and hands it back to Bob. So Bob is convinced that Alice solved the puzzle without giving away the solution.

This idea may be adapted to an identification protocol if each person (or entity) is given a "puzzle" and its answer or indeed each entity makes up a "puzzle" that only they themselves can solve. The security of the system relies on the difficulty of solving the "puzzle". In the case above, if Alice was the only person who could solve a Rubik's cube, then this could be **her** puzzle.

The idea is to associate each person with something unique; something that only that person can reproduce. This, in effect, takes the place of a face or voice, which are unique factors allowing people to identify one another in the physical world. [6]

In general terms, an identification protocol involves a **claimant** *A* and a **verifier** *B*. The verifier is presented with, or presumes beforehand, the purported identity of the claimant. The goal is to corroborate that the identity of the claimant is indeed *A*, i.e. *A* provides entity authentication.

3.1 The Objectives of Identification Schemes

From the point of view of the verifier, the outcome of an identification protocol is either acceptance of the claimant's identity as authentic or rejection of the claimant's identity. More specifically, the objectives of an identification protocol include the following:

1. In the case of honest parties *A* and *B*, *A* is able to successfully authenticate itself to *B*; i.e. *B* will complete the protocol having accepted *A*'s identity.

2. *B* cannot reuse an identification exchange with *A* so as to successfully impersonate *A* to a third party.

3. The probability is negligible that any party *C* distinct from *A*, carrying out the protocol and playing the role of *A*, can cause *B* to complete and accept *A*'s identity.

4. The previous points are true even if a large number of previous authentications between *A* and *B* have been observed or indeed *C* has participated in previous protocol executions with either or both *A* and *B*. [1]

3.2 Applications

There are many common everyday situations where it is necessary to prove "electronically" one's identity.

Typical scenarios include:

- In consumer payment transactions, a token or a card may be presented, bearing an identifier (a code), and a sample signature or an encoded PIN for automated authentication processes.
- To log in remotely to a computer over a network, it suffices to know a valid user name and the corresponding password.
- In establishing a new relationship between a customer and a financial institution, we may need to provide a set of "tokens" such as a passport, driver's licence and/or utility bills bearing the applicant's home address.

It is important to note that in all identification schemes or protocols the phrase "proof of identity" should be avoided in favour of the term "evidence of identity". [22]

3.3 Basis of Identification

Entity authentication techniques may be divided into three main categories, depending on which of the following the security is based:

1. *What you know*.
   Examples include standard passwords, passphrases, PINs and secret or private keys whose knowledge is demonstrated in challenge-response protocols.

2. *What you have.*
   This is normally a physical accessory like a passport or a magnetic-striped card like a credit card. It also includes token-based systems, such as smartcards that have an embedded microprocessor, or password generators that provide time-variant

passwords. Providing something you know – like a password, is usually augmented by the possession of some object, to prove one's identity.

For example, challenge-response systems can include a hand-held token, typically the size of a small calculator, with a keypad. To access a system using challenge-response technology, a user activates the token using a PIN and then enters into the token a series of codes that are challenged by the system. This technology has some limitations however, including the number of user steps, the longer time required to authenticate the identity of a user and the size and complexity of the token.

3. *What you are.*
   This category includes methods that make use of human physical characteristics or involuntary response patterns known as biometrics. These include fingerprints, voice, retinal patterns, hand geometry and face or body profile. [21]

3.4 The Quality of Identification Schemes

In each of the three methods outlined above, there is always the need to strike a balance arising from:

- False rejection (Type I error)
- False acceptance (Type II error)

In a false rejection type error, an identity is accepted that should have been rejected, resulting in the acceptance of imposters and mistaken identity. In a false acceptance type error, an identity is not accepted that should have been, resulting in the rejection of correct matches.

Sources of poor quality identification include:

- Accidental mistakes
- Intentional mistakes, which include masquerading or spoofing (pretending to be a different entity) and the avoidance of identification.

Where quality shortfalls occur, additional considerations come into play which include:

- The **repudiability** of an assertion – for example the question of how an entity contests information stored by another party.
- The **onus of proof** – for example the need to establish on which entity the responsibility lies to establish that data is or is not of appropriate quality. [7]

This balance depends on the extent of protections against abuse and hence if it can be repudiated by the entity concerned. Hence we have the need for different **levels of authentication** that depend on many factors such as our resources, the level of protection that we require, the nature of the communication between the entities involved and the nature of the network communication channels between source and destination.

4. Levels of authentication

There are three main levels of authentication, consisting of:

1. Weak authentication
2. Moderate authentication
3. Strong (zero-knowledge based) authentication

4.1 Weak authentication

Weak authentication may be sub-divided into **one-s**tage authentication and **two-stage** authentication.

4.1.1 One-stage authentication

In the case of one-stage authentication, end users need only one item of information to verify the username they enter when they log on. This one item is usually a password, which often remains the same for a significant amount of time. Furthermore, end users tend to use passwords that are short and easy to remember – usually a family name or some word from a standard dictionary.

Even if users keep their password secret, they maybe captured by protocol analysers or maybe the subject of a **dictionary attack.** Although this brute method approach of finding a particular user's password is not very successful, they are successful at finding passwords in most systems, given the predictable nature of most people's passwords. This success rate is largely due to the **birthday paradox**. The birthday paradox states the counterintuitive result that the probability that two people share the same birthday in a group of 23 people is greater than 0.5. [2]

A further drawback of this method is that it provides no guarantee against privileged insiders or disgruntled employees gaining access to the stored passwords. One way to combat this drawback is to store the passwords in an encrypted file. In this scenario, to verify a user-

entered password, the system computes a **one-way function** of the entered password and compares this to the stored entry for the stated user-id.

Let $f(x)$ be a one-way function. This means it should be easy to compute $f(x)$ but it should be **computationally infeasible** for an opponent to calculate x given the value of y. [4] Generally a problem that cannot be solved in polynomial time is considered infeasible or intractable. Polynomial time in Big-O notation means that the running time of an algorithm with an input of size n is $O(n^t)$ for some constant t. [2]

Another method of making dictionary attacks less effective is that each password, upon initial entry, is randomly padded with an additional 12-bit salt. The salt is a 12-bit random number that is appended to the password resulting in 4,096 possible encryptions of the one password. These 12 bits are then used to modify the function $f(x)$. The result is stored in the password file, along with the user's name and the values of the 12-bit salt. When the user enters the password x, the computer finds the value of the salt for this user in the file, which it then uses in the computation of the modified $f(x)$. This is then compared to the value stored in the file. [5]

Another major weakness of schemes using fixed, reusable passwords is that passwords are transmitted in cleartext over the communications link between the user and the system. An eavesdropper may record this data, thus allowing subsequent impersonation.

4.1.2 Two-stage authentication

Token-based authentication is an example of two-stage authentication. In contrast to password authentication, which relies solely on the use of a single password, two-stage authentication incorporates a PIN in addition to a hardware or software device. Smart cards, which are preprogrammed with unique passwords, are an example of hardware tokens. Another scenario uses a physical device called a token, which generates a token code. The token displays a new code every 60 seconds; therefore each token code is used only once. [14]

Software tokens are generated by software installed on a computer. After being activated by a user upon entering a PIN, a software token provides a unique password for authentication. [15] Hardware tokens that generate One Time passwords seem like a great idea but they are expensive. Additional costs include replacing lost tokens, and administering and updating the authentication server's database. For this reason they have not been widely adopted, even though it's a much more secure system than the use of passwords alone. [16]

4.2 Moderate authentication

A natural progression from fixed password schemes (weak authentication) to challenge-response identification (strong authentication) may be observed by considering **one-time password schemes**. [1] As mentioned earlier, a major security concern of fixed password schemes is eavesdropping and subsequent replay of the password. A partial solution is the use of one-time passwords, where each password is used only once. Such schemes are safe from passive adversaries who eavesdrop and later attempt impersonation.

One variation of this one-time password scheme is the Lamport's scheme, which is based upon one-way functions. Lamport proposed an elegant scheme for one-time passwords requiring no specialised hardware. A general-purpose computer can be used to generate responses based on a secret reusable password. In the scheme, the user's secret password is never sent over an insecure link, where it could be captured by an eavesdropper. Say the secret password is w. We then use a one-way function H. H is used to define a sequence of passwords, w, $H(w)$, $H(H(w))$, etc. A fixed number of identifications are initially agreed to and each password in the sequence is then verified by the intended destination. If each password in the sequence is accepted, then the destination accepts the source as authentic. [10]

This scheme however remains vulnerable to an active adversary who intercepts and traps an as-yet-unused one-time password, for the purposes of subsequent impersonation.

4.3 Strong authentication

As outlined earlier, the idea behind strong authentication techniques or challenge-response protocols is that one entity (the claimant) "proves" its identity to another entity (the verifier) by demonstrating knowledge of a secret only known to itself, without revealing the secret to the verifier during the protocol. This forms the basis of all **zero-knowledge** techniques. The **Zero-Knowledge Identification Protocol (ZKIP)** is part of an **interactive proof system**, which uses zero-knowledge techniques in order to achieve identification.

Informally, an interactive proof is a protocol between two parties in which one party, called the claimant, tries to prove a certain fact to the other party, called the verifier. An interactive proof usually takes the form of a ZKIP protocol (or challenge-response protocol), in which the claimant and the verifier exchange messages and the verifier either "accepts" or "rejects" the fact that the claimant tries to prove. [6]

More formally, the claimant's objective is to prove to the verifier the truth of an assertion, e.g. claimed knowledge of a secret. The verifier either accepts or rejects the proof. However, in an interactive proof system such as this, the traditional mathematical notion of proof where proofs are absolute, needs to be discarded. Instead, interactive proofs are probabilistic rather than absolute and a proof in this context needs to be correct only with bounded probability, albeit possibly arbitrarily close to 1.

Interactive proofs, used in cryptographic applications, have three essential properties:

1. *Completeness*

   The verifier accepts the proof if the claimant knows the fact with an overwhelming probability, and both the claimant and the verifier follow the protocol. The definition of overwhelming probability depends on the application, but it generally implies that the probability of failure is not of practical significance.

2. *Soundness*

   The verifier always rejects the proof, if the claimant does not know the fact, as long as the verifier follows the protocol.

3. *Zero-knowledge*

   The verifier learns nothing about the fact being proved (except that it is correct) from the claimant that he/she could not already learned without the claimant. In a zero-knowledge scheme, the verifier cannot even later prove the fact to anyone else. The essential point here is that only a single bit of information need to be conveyed – namely, that the claimant actually **does know** the fact that it wishes to prove. [8]

5. Zero-Knowledge Identification Protocols (ZKIPs)

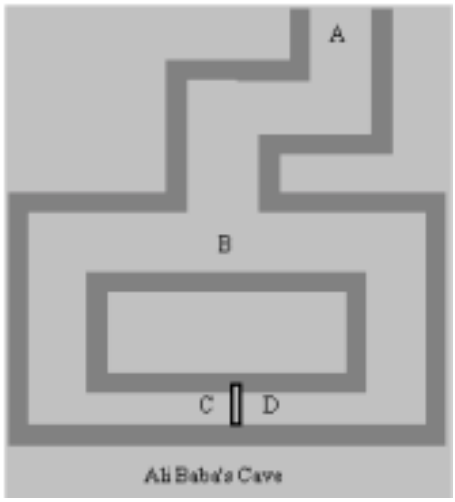
Some years ago, it was reported that some thieves set up a fake ATM machine in a shopping centre. When a person inserted a bankcard and typed in their PIN identification, the machine recorded the information but responded with the message that it could not accept the card. The thieves then made counterfeit bank cards and then went to legitimate ATMs and withdrew cash, using the PIN numbers they had obtained.

How can this be avoided? There are several situations where someone reveals a secret identification number or password in order to complete a transaction. Anyone who obtains

the secret information can masquerade as this person. What is needed is a way to use the secret information without giving away any of this secret information that can be reused by an eavesdropper. This is where zero-knowledge techniques come in.

Quisquater *et al*, [9], explain zero-knowledge protocols by retelling the legendary story of *Ali Baba and the Forty Thieves.*

Alice wants to prove to Bob that she knows the secret words that will open the door at CD in the cave, but she does not wish to reveal the secret to Bob. In this scenario, Alice's commitment is to go to C or D. A typical round in the proof proceeds as follows:



Bob goes to A and waits there while Alice goes to C or D. Bob then goes to B and shouts to Alice to appear from either the right side or the left side of the tunnel. If Alice does not know the secret words "Open Sesame", there is only a 50% chance that she will come out of the side of the tunnel requested by Bob. Bob can repeat this challenge as many times as he desires, until he is certain that Alice knows the secret words. In each round, of course, Alice randomly chooses which side of the tunnel she will go down and Bob randomly chooses which side he will request. Therefore, if Alice comes out the correct side of the tunnel for each of 10 consecutive repetitions, say, there is only one chance in $2^{10} = 1024$ that Alice doesn't know how to go through the door CD.

No matter how many times the proof repeats, Bob will never learn the secret words. Suppose Oscar is watching the proceedings on a video monitor set up at B. He will not be able to use anything he sees to convince Bob or anyone else that he, too, can go through the door. Moreover, he might not even be convinced that Alice can go through the door. After all,

Alice and Bob could have planned the sequence of rights and lefts in advance. By this reasoning, there is no useful information that Bob obtains that can be transmitted to anyone.

Note that there is never a proof, in the strict mathematical sense, that Alice can go through the door. But there is overwhelming evidence (the overwhelming probability referred to earlier) that she can, obtained through a series of challenges and responses. This is, in essence, the nature of zero-knowledge "proofs".

6. Mathematics of Zero-Knowledge Protocols

Zero-knowledge Identification Protocols (ZKIP) are based on **Euler's totient function** and **discrete logarithms** over the **subgroup *Z*/*nZ***. [11]

6.1 Euler's Totient Function

Euler's Totient Function

Euler's Totient function is written as $\phi(n)$, where $\phi(n)$ is the number of positive integers less than and **relatively prime** to *n*.

Definition

The integers *a* and *b* are relatively prime if they have no prime factors in common, that is, if their only common factor is 1. This is equivalent to saying that *a* and *b* are relatively prime if gcd (*a*,*b*) = 1 where gcd (*a*, *b*) stands for the Greatest Common Divisor of *a* and *b*.

Example:

$\phi(21) = 12$ where the 12 integers are {1, 2, 4, 5, 8, 10, 11, 13, 16, 17, 19, 20}

It follows from the above that for a prime number *p*

$$\phi(p) = p - 1$$

Now suppose that we have two prime numbers *p* and *q*. Then for *n* = *pq*

$$\phi(pq) = (p-1)(q-1)$$

Example:

21 = (3)(7)

So $\phi(21) = (3 - 1)(7 - 1) = 12$ and the 12 integers are listed above.

6.2 Congruences

Definition

Let *a*, *b*, *n* be integers.

We say that $a \equiv b \mod n$

i.e. *a* is congruent to *b* mod *n* if *a* – *b* is a multiple of *n*.

6.3 The Extended Euclidean Algorithm

Theorem

$b \in Z_n$ has a multiplicative inverse if and only if gcd (*b*, *n*) = 1

The set of residues modulo *n* that are relatively prime to *n* is denoted by Z_n^*. Any element in Z_n^* will have a multiplicative inverse b^{-1}, which is also in Z_n^*.

The extended Euclidean algorithm is an efficient way to compute b^{-1}. [24]

Example:

Compute $28^{-1} \mod 75$

Solution:

$75 = 2 \times 28 + 19$

$73 \times 28 \mod 75 = 19$

$28 = 1 \times 19 + 9$

$3 \times 28 \mod 75 = 9$

$19 = 2 \times 9 + 1$

$67 \times 28 \mod 75 = 1$

$9 = 9 \times 1$

Hence, $28^{-1} \mod 75 = 67$

Proposition

Suppose gcd (*a*, *n*) = 1.

Let *s* and *t* be integers such that *as* + *nt* = 1. Integers *s* and *t* can be found using the Extended Euclidean algorithm.

Then $as \equiv 1 \mod n$, so *s* is the multiplicative inverse for *a* (mod *n*).

6.3.1 Solving $ax \equiv c \mod n$ when gcd (a, n) = 1

1. Use the Extended Euclidean algorithm to find integers s and t such that $as + nt = 1$.
2. The solution is $x \equiv cs \mod n$.

Example:

Solve $11111x \equiv 4 \mod 12345$

Solution:

Using the Extended Euclidean algorithm, we find that

$$11111.2471 + 12345.\ t = 1$$

Hence $11111.2471 \equiv 1 \mod 12345$

Multiplying each side by 2471 yields

$$x \equiv 9884 \mod 12345.$$

In practice, this means that if we are working mod 12345 and we meet the fraction $\frac{4}{11111}$, we can replace it with 9884.

6.4 Chinese Remainder Theorem

Suppose gcd (m, n) = 1. Given a and b, there exists exactly one solution x (mod mn) to the simultaneous congruences $x \equiv a \mod m$ and $x \equiv b \mod n$

Example:

Solve $x \equiv 3 \mod 7$ and $x \equiv 5 \mod 15$

Solution:

$x \equiv 80 \mod 105$ since 105 = 7.15

Since $80 \equiv 3 \mod 7$ and $80 \equiv 5 \mod 15$, 80 is a solution. [23]

6.5 Modular Exponentiation

We now consider numbers in the form $x^a \mod n$.

Example:

Suppose we want to compute $2^{1234} \mod 789$.

Solution:

We could perform each multiplication and then work out the remainder. This method is too slow to be of any practical value, so instead we begin with $2^2 \equiv 4 \bmod 789$ and repeatedly square both sides as follows:

$$2^4 \equiv 4^2 \equiv 16$$
$$2^8 \equiv 16^2 \equiv 256$$
$$\vdots$$
$$\vdots$$
$$\vdots$$
$$2^{1024} \equiv 286$$

Since 1234 = 1024 + 128 + 64 + 16 + 2

we have $2^{1234} \equiv 286.559.367.49.4 \equiv 481 \bmod 789$.

6.6 Fermat's Little Theorem

Fermat's Little Theorem

If *p* is a prime and *p* does not divide *a*, then $a^{p-1} \equiv 1 \bmod p$

Example:

Compute $2^{43210} \bmod 101$

Solution:

From Fermat's Theorem, we know that $2^{100} \equiv 1 \bmod 101$.

Therefore, $2^{43210} \equiv (2^{100})^{432} 2^{10} \equiv 1^{432} 2^{10} \equiv 1024 \equiv 14 \bmod 101$.

6.7 Euler's Theorem

Euler'sTheorem

For every *a* and *n* that are relatively prime, $a^{\phi(n)} \equiv 1 \bmod n$

Example:

If $a = 3$ and $n = 10$,

$\phi(10) = 4$ and $3^4 = 81 \equiv 1 \bmod 10$

Definition

Z_n^* denotes the set of numbers *i*, $0 \le i \le n$, which are relatively prime to *n*.

Example:

$Z_9^* = \{1, 2, 4, 5, 7, 8\}$

Multiplication Table

* mod 9	1	2	4	5	7	8
1	1	2	4	5	7	8
2	2	4	8	1	5	7
4	4	8	7	2	1	5
5	5	1	2	7	8	4
7	7	5	1	8	4	2
8	8	7	5	4	2	1

Theorem

Z_n^* forms a group under modulo *n* multiplication. The identity element is *e* = 1.

Definition

The order of an element $a \in (G,\circ)$ is the smallest positive integer such that $a \circ a \circ ..... \circ a = a^0 = 1$ where $\circ$ is some binary operation defined on the group *G*.

So, for example, from the multiplication table above, the order of *a* = 2 in $(Z_9^*,\times)$ is 6 because $2^6 = 1$, i.e. ord (2) = 6.

In fact, we know that if *p* is prime, then Z_p^* is a group of order *p* – 1.

Definition

A group $(G,\circ)$ which contains elements *a* with maximum order ord (*a*) = | *G* | is said to be **cyclic**. Elements with maximum order are called **generators** or **primitive elements (roots)** of the group $(G,\circ)$.

Note: |*G*| is the number of elements in the group *G*. [23]

This, in effect, implies that if *p* is prime, then the group Z_p^* is in fact cyclic: there exists an element $a \in Z_p^*$ having order equal to *p* – 1.

Theorem

If *p* is prime, then Z_p^* is a cyclic group.

Example:

Suppose $p = 13$.

Then 2 is a primitive element modulo 13 because

$2^i \bmod 13 = 2, 4, 8, 3, 6, 12, 11, 9, 5, 10, 7, 1$ where $1 \le i \le 12$.

i.e. 2 generates all 12 elements of Z_{13}^*.

The element 2^i is primitive if and only if gcd (i, 12) = 1, i.e., $i = 1, 5, 7$ or 11.

Hence, the primitive elements modulo 13 are 2, 6, 7 and 11.

More generally, we can say that the highest possible exponent to which a number can belong (mod *n*) is $\phi(n)$. If a number is of this order, it is referred to as a primitive element of *n*,

i.e. $a, a^2, ..., a^{\phi(n)}$

are distinct (mod *n*) and are all relatively prime to *n*.

This implies that $a^{\phi(n)} \equiv 1 \mod n$……. Euler's Theorem

6.8 Square Roots Mod *n*

How do we find the solution or solutions to $x^2 \equiv 71 \mod 77$? Or more generally, consider the problem of finding all the solutions of $x^2 \equiv b \mod n$, where $n = pq$ is the product of two primes. It can be shown that this can be done quite easily once the factorisation of *n* is known. Conversely, if we know all the solutions, then it is easy to factor *n*.

Proposition

Let $p \equiv 3 \mod 4$ be prime and let *y* be an integer. Let $x \equiv y^{(p+1)/4} \mod p$

1. If *y* has a square root mod *p*; then the square roots of *y* mod *p* are $\pm x$.
2. If *y* has no square root mod *p*, then $-y$ has a square root mod *p*, and the square roots of $-y$ are $\pm x$.

Example 1:

Solve the equation $x^2 \equiv 5 \mod 11$

Solution:

Since $(p + 1)/4 = 3$,

we compute $x \equiv 5^3 \equiv 4 \mod 11$.

Since $4^2 \equiv 5 \mod 11$,

$x = \pm 4$.

Example 2:

Solve the equation $x^2 \equiv 71 \mod 77$

Solution:

$x^2 \equiv 71 \equiv 1 \mod 7$ and $x^2 \equiv 71 \equiv 5 \mod 11$.

Therefore, $x \equiv \pm 1 \mod 7$ and $x \equiv \pm 4 \mod 11$.

Using the Chinese Remainder Theorem, we can combine a congruence mod 7 and a congruence mod 11 into a congruence mod 77.

Here, we recombine in four ways to get the solutions $x \equiv \pm 15, \pm 29 \mod 77$

It follows from the above that if $a \equiv b \mod p$ and $a \equiv -b \mod q$, gcd $(a - b, n) = p$ and we have found a nontrivial factor of $n = pq$. In the example above, we know that $15^2 \equiv 29^2 \equiv 71 \mod 77$. Therefore, gcd (15 – 29, 77) = 7 gives a nontrivial factor of 77.

We can now state in summary an important result:

> Suppose $n = pq$ is the product of two primes congruent to 3 mod 4, and suppose y is a number relatively prime to n which has a square root mod n.
>
> Then finding the four solutions $x \equiv \pm a, \pm b$ to $x^2 \equiv y \mod n$ is computationally equivalent to factoring n.

6.9 Finite Fields

Loosely speaking, a set that has the operations of addition, multiplication, subtraction and division by nonzero elements is called a field.

Examples of fields include real numbers, complex numbers and the integers mod a prime number, i.e. Z_p. But the set of all integers is **not** a field because we sometimes cannot divide and obtain an answer in the set, e.g. 4/3 is not an integer. [25]

6.10 The Discrete Logarithm Problem

We begin by describing the problem in the setting of a finite field Z^p, where p is prime. The problem is considered to be difficult if p is carefully chosen.

In particular, there is no known polynomial-time algorithm for the Discrete Logarithm problem. To thwart known attacks, *p* should be at least 150 digits and *p* – 1 should have at least one "large" prime factor. [3]

The main advantage of the Discrete Logarithm problem in challenge-response protocols (and indeed in cryptography generally) is that finding discrete logs is difficult, but the inverse operation of exponentiation can be computed efficiently. Stated another way, exponentiation modulo *p* is a one-way function for suitable primes *p*, i.e. it is computationally infeasible.

The Discrete Logarithm problem can be formally stated as follows:

We assume that *p* is prime and α is a primitive element modulo *p*. We take *p* and α to be fixed. From the foregoing discussion, we know that the powers of α from 1 through to (*p* – 1) produce each integer from 1 through (*p* – 1) exactly once.

It follows, therefore, that given $\beta \in Z_p^*$, we can find the unique exponent *a*, $0 \le a \le p-1$, such that $\alpha^a \equiv \beta (\mathrm{mod}\ p)$. In other words, taking logs, we have $a = \log_\alpha(\beta)$

The problem of finding ***a*** is called the Discrete Logarithm problem. Note that if we dispensed with the requirement that α be a primitive root, then the discrete logarithm will not be defined for certain values of β.

7. Identification Protocols

7.1 General structure of Zero-Knowledge Protocols

The Fiat-Shamir protocol illustrates the general structure of a large class of **three-move** zero-knowledge protocols:

$A \rightarrow B$: witness
$B \rightarrow A$: challenge
$A \rightarrow B$: response

The design of these protocols ensures that only the legitimate party *A*, with knowledge of *A*'s secret, is truly capable of answering all the questions. Furthermore, the answer to any of these questions provides no information about *A*'s long-term secret.

A responds to at most one challenge (question) for a given witness, and should not reuse any witness. [1]

7.2 Fiat-Shamir Identification Protocol

What follows is a basic version of the Fiat-Shamir Zero-Knowledge protocol. The objective is for Alice to identify herself by proving knowledge of a secret *s* to any verifier such as Bob or a trusted centre called Tom, without revealing any information about *s* not known or computable by Bob or Tom prior to execution of the protocol. The security of the method relies on the difficulty of extracting square roots, modulo large composite integers *n* of unknown factorisation, which is equivalent to factoring *n*.

The protocol involves the following steps:

1. *One-time setup*

   (i) A trusted authority – call it Tom – selects two large prime numbers *p* and *q*, calculates $n = pq$, publishes *n*, and keeps the primes secret.

   (ii) Alice, a user of the centre, selects a number *s* in the range 1 to $n - 1$ that is relatively prime to *n*, calculates $v = s^2 \bmod n$, registers *v* as her public key with Tom, keeps her password *s* secret and carries both *s* and *v* with her.

   (iii) Alice and Bob agree on a maximum number of rounds *t* of the identification protocol that will be carried out at login.

2. *Protocol messages*

   Each of *t* rounds has three messages as follows:

   Alice $\rightarrow$ Bob: $x = r^2 \bmod n$

   Bob $\rightarrow$ Alice: $e \in \{0, 1\}$

   Alice $\rightarrow$ Bob: $y = r.s^e \bmod n$

3. *Protocol actions*

   The following steps are iterated *t* times – sequentially and independently. Bob accepts the proof if all *t* rounds succeed.

   (i) Alice first generates a random number *r* in the range 1 to $n - 1$. This is called her **commitment.** She calculates $x = r^2 \bmod n$ - called her **witness** – and sends *x* to Bob.

   (ii) On receiving *x*, Bob selects a bit *e*, either 0 or 1, at random – called a challenge and sends *e* to Alice.

(iii) On receiving *e*, Alice calculates $y = r.s^e \bmod n$. If $e = 0$ then $y = r$ and if $e = 1$, $y = r.s \bmod n$.

y is called the **response.** She sends *y* to Bob.

(iv) Receiving y, Bob calculates

$$z = y^2 \bmod n$$

and

$$z' = x.v^e \bmod n$$

If $z \neq z'$, he refuses the login from Alice; if $z = z'$, he accepts the round of the protocol. [1]

In the later case, if fewer than *t* rounds have been carried out, Alice starts a new round by picking another number at random and if *t* rounds are all successful, Alice's identification is complete and she is logged in.

The following congruences show that Alice will be able to prove her identity to Bob:

$$\begin{aligned} xv^e &\equiv r^2 v^e \bmod n \\ &\equiv r^2 s^{2e} \bmod n \\ &\equiv (rs^e)(rs^e) \bmod n \\ &\equiv y^2 \bmod n \end{aligned}$$

So Bob will accept Alice's proof of identity.

7.3 Example to illustrate the Fiat-Shamir Protocol

1. *One-time setup*

(i) Suppose $p = 37$ and $q = 101$; then $n = 37.101 = 3737$.

(ii) If Alice selects $s = 113$ (which is relatively prime to *n*), then her public key is $v = s^2 \bmod 3737 = 113^2 \bmod 3737 = 1558$.

(iii) Bob and Alice now agree on $t = 8$ rounds of the protocol. This completes the setup.

2. *Protocol Actions*

(i) Now Alice wants to login from a remote location. She chooses a number at random: $r = 3284$ (her commitment).

Then her witness is $x = 3284^2 \bmod 3737 = 3411$.

She sends her public key 1558 and her witness 3411 to Bob.

(ii) He receives these values and flips a coin to determine the challenge

e = 1. He sends 1 to Alice.

(iii) On receiving e = 1,

Alice calculates $y = 3284.113^1 = 1129 \pmod{3737}$.

She sends this back to Bob (the response).

(iv) He then verifies $z = y^2 \bmod 3737 = 1129^2 \bmod 3737 = 324$

and $z' = x.v^e \bmod 3737 = 3411.1558 \bmod 3737 = 324$.

Because these two values are the same, Bob accepts the first round of the protocol. Say z and z' agree over 8 rounds of the protocol. Bob estimates that the probability this person is **not** Alice is

$$\left(\frac{1}{2}\right)^8 \approx 0.0039 .$$

From Bob's perspective, the probability that it **is** Alice is 1 – 0.0039 = 0.9961.

7.4 Security of Fiat-Shamir Protocol

An adversary, Oscar, could try guessing Bob's challenge for each round of the protocol. So if Oscar can guess Bob's challenge correctly for each of t rounds of the protocol, he can fool Bob into believing that Oscar is in fact Alice. The chances of Oscar completing t rounds of the protocol successfully is

$$\frac{1}{2}.\frac{1}{2}.\frac{1}{2}.....\frac{1}{2}(t \text{ times}) = \left(\frac{1}{2}\right)^t .$$

With t = 20 rounds, the probability of Oscar succeeding in impersonating Alice is 0.000976563 – an extremely unlikely event! Now suppose that Oscar listens in on the protocol between Alice and Bob. Can he infer from it the value of s? At each round, he will only see v, e and y.

In round 1, in our example, because e = 1, Oscar will know that $r.s$ = 1129 (mod 3737) and r^2 = 3411 (mod 3737).

He could calculate the square root of 3411 mod 3737 by trail and error, find r and solve for s. With such a small value of n as in this example, this would pose little difficulty. But in real implementations of the protocol, n will be of the order of 200 decimal digits and then the square root problem becomes **intractable**. So the effectiveness of the protocol depends on the purported intractability of the square root problem modulo pq (= n).

7.5 Feige-Fiat-Shamir (FSS) Identification Protocol

The Fiat-Shamir protocol described above can be generalised and the FSS protocol is an example of one such generalisation.

In summary, Alice has secret numbers $s_1, s_2, ..., s_k$.

Let $v_i \equiv s_i^{-2} \pmod{n}$, where we assume gcd (s_i , n) = 1.

The numbers v_i are sent to Bob. Bob will try to verify that Alice knows the numbers

$$s_1, s_2, ..., s_k .$$

The protocol involves the following steps:

1. *One-time setup*:
   (i) A trusted authority – call it Tom – selects two large prime numbers *p* and *q*, congruent to 3 mod 4, calculates $n = pq$, publishes *n*, and keeps the primes secret. Tom also publishes the parameters *k*, the key size and *t*, the number of protocol iterations.
   (ii) Alice, a user of the centre, selects *k* secret random numbers $s_1, s_2, ..., s_k$ in the range $1 \le s_i \le n-1$ where gcd (s_i , n) = 1.
   This ensures that *n* cannot be factored easily.
   (ii) Alice computes $v_i \equiv s_i^{-2} \pmod{n}$ and registers her public key $(v_1, ..., v_k; n)$, while only Alice knows her private key ($s_1, s_2, ..., s_k$) and *n*.
2. *Protocol messages*
   Each of *t* rounds has 3 messages as follows:
   Alice $\rightarrow$ Bob : $x \equiv r^2 \pmod{n}$
   Bob $\rightarrow$ Alice : $(e_1, ..., e_k)$, $e_i \in \{0, 1\}$
   Alice $\rightarrow$ Bob : $y \equiv r.s_1^{e_1} s_2^{e_2} .....s_k^{e_k} \pmod{n}$
3. *Protocol actions*
   (i) Alice chooses a random number *r* (the commitment), computes
   $x \equiv r^2 \pmod{n}$ (the witness) and sends *x* to Bob.
   (ii) Bob chooses numbers $(e_1, ..., e_k)$, $e_i \in \{0, 1\}$. He sends these to Alice.
   (iii) Alice computes $y \equiv r.s_1^{e_1} s_2^{e_2} .....s_k^{e_k} \pmod{n}$ and sends *y* (the response) to Bob.
   (iv) Bob checks that $x \equiv y^2 v_1^{e_1} v_2^{e_2} .....v_k^{e_k} \pmod{n}$
   (v) Steps (i) through (iv) are repeated *t* times.

The following congruences show that Alice will be able to prove her identity to Bob:

$$x \equiv y^2 v_1^{e_1} v_2^{e_2} \ldots\ldots\ldots v_k^{e_k} \mod n$$
$$\equiv (r^2 s_1^{2e_1} \ldots\ldots\ldots s_k^{2e_k})(v_1^{e_1} \ldots\ldots\ldots v_k^{e_k}) \mod n$$
$$\equiv (r^2 s_1^{2e_1} \ldots\ldots\ldots s_k^{2e_k})(s_1^{-2e_1} \ldots\ldots\ldots s_k^{-2e_k}) \mod n$$
$$\equiv r^2 \mod n$$

So Bob will accept Alice's proof of identity.

7.6 Example to illustrate the FFS Protocol

1. *One-time setup*

(i) Tom (the trusted centre) selects the primes p = 683, q = 811 and publishes n = pq = 553913. Integers k = 3 and t = 1 are defined as security parameters.

(ii) Alice selects 3 random numbers s_1 = 157, s_2 = 43215, s_3 = 4646.

(iii) Alice computes $v_1 \equiv s_1^{-2} \pmod n$ = 441845, v_2 =338402 and v_3 = 124423.

Alice's public key is (441845, 338402, 124423; 553913) and her private key is (157,43215, 4646).

2. *Protocol actions*

(i) Alice chooses r = 1279 (the commitment), computes $x = r^2 \mod n$ = 25898 and sends x to Bob.

(ii) Bob sends to Alice (the challenge) the 3-bit vector (0, 0, 1).

(iii) Alice computes $y \equiv r.s_1^{e_1} s_2^{e_2} \ldots\ldots\ldots s_k^{e_k} = r.s_3^1 \mod n$ = 403104 and sends y (the response) to Bob.

(iv) (iv) Bob computes $z \equiv y^2 v_1^{e_1} v_2^{e_2} \ldots\ldots v_k^{e_k} \pmod n = y^2.v_3^1 \mod n$ = 25898 and accepts Alice's identity since $z = x$. [1]

7.7 Security of Feige-Fiat-Shamir (FFS) Protocol

The security of the FFS protocol relies on the difficulty of extracting square roots modulo n. This is equivalent to factoring n. The best attack, using a chosen message, has a probability 2^{-kt} of successful impersonation.

An adversary, Oscar, who doesn't know the numbers $s_1, s_2, ..., s_k$, could try and guess the string of bits $(e_1, ..., e_k)$ that Bob will send. He lets *y* be a random number and declares $x \equiv y^2 v_1^{e_1} v_2^{e_2} .....v_k^{e_k} \pmod{n}$. When Bob sends the string of bits, Oscar sends back the value of *y*.

For example, suppose Oscar can guess the correct response when $e_1 = 1$, e_2 = 1, e_4 = 1 and all other e_i = 0. However, suppose Bob sends $e_1 = 1$, e_3 = 1 and all other e_i = 0. Then Oscar, posing as Alice, will be ready to supply a square root of $xv_1v_2v_4$, but will be asked to supply a square root of xv_1v_3. This, combined with what he already knows, is equivalent to knowing a square root of $v_2^{-1}v_3v_4^{-1}$, which he is **not** able to compute.

In effect, there are 2^k possible strings of bits that Bob can send to fool Oscar. In one iteration of the protocol, the chances are only one in 2^k that Bob will be fooled. If the procedure is repeated *t* times, the chances are one in 2^{kt} that Oscar will be fooled.

Recommended values are *k* = 5 and *t* = 4, which gives the same probability as 20 iterations of the previous Fiat-Shamir scheme or 1 in a million chance of impersonation. So the FFS protocol is more efficient in terms of communication between Alice and Bob.

FFS is a pretty simple, and effective zero-knowledge proof. There is, however, an important security tradeoff that needs to be addressed. If you set *t* = 1, computation and communications can be reduced. Also, while holding *kt* constant, and incrementing *k*, while decrementing *t*, will result in the protocol no longer being able to hold the concept of *proof of knowledge*. This means that as *t* approaches 1, the protocol become less and less *sound*. [17]

7.8 FFS as an Identity-Based Scheme

Let *S* be a string that includes Alice's name, address and date of birth. Let *f* be a one-way function (a public **hash** function, for example). A trusted authority, Tom (could be a bank, for example), chooses *n* = *pq* as before and then computes Alice's public values

$$v_i = f(S, i),\ 1 \le i \le k.$$

Now, Tom, knowing the factorisation of *n*, computes a square root s_i for each v_i, and gives these to Alice. Tom can now discard $s_1, s_2, ..., s_k$ and the values of *p* and *q*. This adds to

the security of the scheme since someone who breaks into Tom's computer cannot compromise Alice's security.

Say Alice goes to an ATM for example. The ATM reads *S* from Alice's card. It downloads $(v_1,.....,v_k;n)$ from a database. The FFS protocol is then performed to verify that Alice knows $s_1, s_2,....., s_k$. After a few iterations, the ATM is convinced that the person is in fact Alice and allows her to withdraw cash.

To avoid a lot of typing on Alice's part, a better implementation would be to use chips embedded in the card and store the data in such a way that it cannot be extracted. [5]

7.9 Schnorr Identification Protocol

1. *Setup*

   A trusted authority – Tom – chooses the following parameters:

   (i) p is a large prime such that the discrete logarithm problem in Z_p^* is intractable.

   (ii) q is a large prime divisor of $p - 1$.

   (iii) $\alpha \in Z_p^*$ has order q, i.e. if β is a primitive element mod p, then $\alpha = \beta^{(p-1)/q} \bmod p$.

   (iv) A security parameter t such that $q > 2^t$. For most applications, t = 40 provides adequate security.

   (v) Tom also establishes a secure **signature scheme** with a secret signing algorithm sig_{Tom} and a public verification algorithm ver_{Tom}.

   (vi) A secure **hash function** is specified. The hash function is used to hash the message before it is signed.

2. *Issuing a certificate to Alice*

   (i) Tom establishes Alice's identity by means of conventional forms of identification such as birth certificate or passport. Then Tom forms a string ID (Alice) which contains her identity information.

   (ii) Alice secretly chooses a random exponent a, where $0 \le a \le q-1$. She then computes $v = \alpha^{-a} \bmod p$ and gives v to Tom.

   (iii) Tom generates a signature $s = sig_{Tom}(\mathrm{ID}(\mathrm{Alice}), v)$

   and gives the certificate $cert_{Alice}$ = (ID(Alice), v, s) to Alice.

3. *Protocol messages*

Alice $\rightarrow$ Bob : $cert_{Alice}$ and $x = \alpha^r \bmod p$

Bob $\rightarrow$ Alice: $e, 1 \le e \le 2^t$

Alice $\rightarrow$ Bob: $y = r + ae \bmod q$

4. *Protocol actions*

   (i) Alice chooses a random number r (the commitment), where $0 \le r \le q-1$. She then calculates (the witness) $x = \alpha^r \bmod p$ and sends her certificate $cert_{Alice}$ = (ID(Alice), v, s) and x to Bob.

   (ii) Bob verifies the signature of Tom by checking that ver_{Tom} (ID(Alice), v, s) = true

   (iii) Bob chooses a random number e (the challenge), $1 \le e \le 2^t$ and sends it to Alice.

   (iv) Alice computes $y = r + ae \bmod q$ and sends y to Bob.

   (v) Bob verifies that $z \equiv \alpha^y v^e \bmod p$ and accepts Alice's identity if $z = x$.

For discrete logarithms to be computationally infeasible (or intractable), we require that $p \ge 2^{1024}$ and $q \ge 2^{160}$.

There are two things happening in this identification protocol: Firstly, the signature s proves the validity of Alice's certificate. So Bob verifies the signature of Tom on Alice's certificate to convince himself that the certificate itself is authentic. Secondly, the value a functions like a PIN in that it convinces Bob that the person carrying out the protocol is indeed Alice.

The following congruences show that Alice will be able to prove her identity to Bob:

$$\begin{aligned} \alpha^y v^e &\equiv \alpha^{r+ae} v^e \bmod p \\ &\equiv \alpha^{r+ae} \alpha^{-ae} \bmod p \\ &\equiv \alpha^r \bmod p \\ &\equiv x \bmod p \end{aligned}$$

So Bob will accept Alice's proof of identity.

The Schnorr scheme is designed to be very fast and efficient, both from a computational viewpoint and the amount of information that needs to be exchanged in the protocol. It is also designed to minimise the amount of computation done by Alice. This makes it quite attractive in applications where Alice can use a smart card and where Bob needs to perform more

complex computations. [3] To illustrate the point above, assume that ID (Alice) is a 512-bit string, *v* is also 512 bits and *s* will be 320 bits if the Digital Signature Standard (DSS) is used as a signature scheme. The total size of the certificate which needs to be stored on Alice's card is then 1344 bits. Now we can calculate the number of bits that are communicated during the protocol. Recall the 3 steps of the protocol.

Alice $\rightarrow$ Bob : $cert_{Alice}$ and $x = \alpha^r \bmod p$

Bob $\rightarrow$ Alice: $e, 1 \leq e \leq 2^t$

Alice $\rightarrow$ Bob: $y = r + ae \bmod q$

Alice sends to Bob 1344 + 512 = 1856 bits of information in step 1.
Bob sends Alice 40 bits in step 2.
Alice sends 140 bits in step 3.
So the communication requirements are quite modest. [3]

The computations performed by Alice require the modular exponentiation $x = \alpha^r \bmod p$, which, although computationally intensive, can be performed offline. The computation of $y = r + ae \bmod q$ comprises one modular addition and one modular multiplication, which is not computationally intensive.

On the other hand, Bob's calculations are computationally intensive, since he has to verify Tom's signature on Alice's certificate and also verify that $z \equiv \alpha^y v^e \bmod p \equiv x$. A hash function is a one-way function that produces a message digest of the entire message. The message digest is combined with Alice's secret key to produce Alice's digital signature. [18]

7.10 Example of Schnorr Identification Protocol

1. *Setup*
   (i) Suppose p = 88667, q = 1031 and $p - 1$ is divisible by the prime q.
   The element α = 70322 has order q in Z_p^*, where $\alpha = \beta^{(p-1)/q} \bmod p$ and β is a primitive element of Z_p^*.
   (ii) Suppose Alice's secret exponent is a = 755 then
   $$v = \alpha^{-a} \bmod p$$
   $$= 70322^{1031-755} \bmod 88667$$
   $$= 13136$$
2. *Protocol actions*

(i) Now Alice chooses $r = 543$. She then computes:

$$x = \alpha^r \bmod p$$

$$= 70322^{543} \bmod 88667$$

$$= 84109.$$

She sends x to Bob.

(ii) Bob sends the challenge $e = 1000$ to Alice.

(iii) Now Alice computes $y = r + ae \bmod q$

$$=543 + 755.1000 \bmod 1031$$

$$= 851$$

and sends y to Bob.

(iv) Bob then verifies that

$$x \equiv z \equiv \alpha^y v^e \bmod p$$

i.e. $84109 \equiv 70322^{851}13136^{1000} \bmod 88667.$

and accepts Alice's identity if $z \equiv x$.

7.11 Security of Schnorr Protocol

While it is our hope that an adversary, Oscar, will not gain any information about a when Alice proves her identity (zero-knowledge property), the Schnorr identification protocol has not been proven **secure**. The protocol is not secure for large e, because through interaction, Bob obtains the solution (x, y, e) to the equation $\alpha^y v^e \equiv x \bmod p$, which Bob himself might not be able to compute.

But a modification to the Schnorr scheme was designed by Okamoto, which can be proven to be secure. The main difference between the two schemes is that instead of Tom choosing

$$\alpha \in Z_p^* \text{ of order } q$$

as in the Schnorr scheme, Tom instead chooses two elements

$$\alpha_1, \alpha_2 \in Z_p^*, \text{ both of order } q.$$

Tom keeps the value $c = \log_{\alpha_1} \alpha_2$ secret from all participants including Alice, which we assume is infeasible for any adversary to compute. [3]

Although an adversary, Oscar, could gain access to Alice's correct certificate (since the information on a certificate is revealed each time the identification is run), he will not be able to impersonate Alice unless he knows the value of a. Oscar would have to compute y for each round, but y is a function of a. The computation of a from v involves a discrete logarithm problem, which we assume is intractable.

7.12 Application of Schnorr Identification Scheme

Stefan Brand [18] has developed a system of **digital cash**, which uses the Schnorr identification protocol twice. Digital cash refers to electronic records or messages which serve as money and can be authenticated by the institution granting the digital cash. Essentially, it is a payment message bearing a digital signature that functions as a medium of exchange. [19]

When a customer (Alice) withdraws a "coin" from a bank, the bank binds the user's identity to the "coin", but sends along additional information that allows the customer to **blind** the signed "coin" as seen by the bank. Blinding is the process by which a bank cannot identify the person who withdrew the "coin". While this maintains the customer's identity, it poses the problem for the bank of identifying double-spenders.

Alice challenges the bank to prove knowledge of its secret key. This verifies that the bank has provided a valid signature on the "coin". When Alice spends the blinded "coin", the merchant challenges her to provide knowledge of her secret key. The merchant records the challenge and response and gives this to the bank as part of the "coin" deposit protocol. If the bank receives the same "coin" twice, the challenge and response will reveal Alice's identity if Alice was responsible for double-spending. [20]

7.13 Guillou-Quisquater (GQ) Identification Protocol

The GQ protocol is an extension to the Fiat-Shamir protocol. It allows a reduction in both the number of messages exchanged and memory requirements for user secrets. Furthermore, like the Fiat-Shamir scheme, it is suitable for applications in which the claimant has limited power and memory.

Alice proves her identity to Bob in a 3-pass protocol.

1. *Setup*
   (i) A trusted authority – call it Tom – selects two large prime numbers p and q, and calculates $n = pq$.
   (ii) Tom defines a large prime integer b that functions as a security parameter.
   (iii) Alice secretly chooses an integer u, where $0 \le u \le n-1$. Alice computes $v = u^{-b} \bmod n$ and gives v to Tom.

   The trusted authority, Tom, establishes Alice's identity (as in the Schnorr scheme) and issues the identification string ID (Alice).

The certificate $cert_{Alice}$ = (ID(Alice), *v*, *s*) is given to Alice where $s = sig_{Tom}$(ID(Alice),*v*).

2. *Protocol messages:*

The protocol involves 3 messages:

Alice $\rightarrow$ Bob : $cert_{Alice}$ and $x = r^b \bmod n$

Bob $\rightarrow$ Alice:*e*, $1 \le e \le b-1$

Alice $\rightarrow$ Bob: $y = ru^e \bmod n$

3. *Protocol actions:*

(i) Alice chooses a random *r* (the commitment), $0 \le r \le n-1$ and computes

(ii) $x = r^b \bmod n$ and sends her certificate $cert_{Alice}$ = (ID(Alice), *v*, *s*) and *x* to Bob.

(iii) Bob verifies the signature of Tom by checking that

ver_{Tom} (ID(Alice), *v*, *s*) = true

(iv) Bob chooses a random number *e*, $1 \le e \le b-1$ and sends it to Alice.

(v) Alice computes $y = r\,u^e \bmod n$ and sends it to Bob.

(vi) Bob verifies that $x \equiv v^e y^b \bmod n$.

The following congruences show that Alice will be able to prove her identity to Bob:

$$\begin{aligned} v^e y^b &\equiv u^{-be}(ru^e)^b \bmod n \\ &\equiv u^{-be} r^b u^{eb} \bmod n \\ &\equiv r^b \bmod n \\ &\equiv x \bmod n \end{aligned}$$

So Bob will accept Alice's proof of identity.

7.14 Example of Guillou-Quisquater (GQ) Identification Protocol

1. *Setup*

(i) The trusted authority, Tom, selects primes *p* = 467, *q* = 479 and computes

(ii) *n* = *pq* = 223693.

(iii) Suppose also that *b* = 503 and Alice's secret integer *u* = 101576.

Alice computes

$v = u^{-b} \bmod n$

$= 101576^{-503} \mod 223693$

$= 89888$

and gives this value to Tom.

2. *Protocol actions*

(i) Alice selects $r = 187485$ and computes

$x = r^b \mod n$

$= 187485^{503} \mod 223693$

$= 24412.$

(ii) Alice now sends $cert_{Alice}$ and $x = 24412$ to Bob.

(iii) Bob verifies the signature of Tom on Alice's certificate by checking that ver_{Tom} (ID(Alice), v, s) = true.

(iv) Now Bob sends a random challenge $e = 375$.

(vii) Alice replies with $y = r.u^e \mod n$

$= 187485.101576^{375} \mod 223693$

$= 93725$

and sends it to Bob.

(viii) Bob then verifies that

$x \equiv v^e y^b \mod n$

i.e. $24412 \equiv 89888^{375}93725^{503} \mod 223693.$

Hence Bob accepts Alice's proof of identity.

7.15 Security of Guillou-Quisquater Protocol

Extracting b^{th} roots modulo the composite integer n is necessary to defeat the protocol; this is no harder than factoring n, which we already know to be computationally intractable.

8. Comparison of Fiat-Shamir, Schnorr and Guillou-Quisquater Protocols

Each of these protocols provides solutions to the identification problem. Each has relative advantages and disadvantages with respect to various performance criteria and for specific applications.Each protocol can be compared under the following criteria:

1. *Computational efficiency*

Fiat-Shamir requires from 11 to about 30 modular multiplications or steps by the claimant (Alice) with $kt = 20$ and n is 512 bits while GQ requires about 60 steps.

2. Offline computations

The Schnorr scheme has the advantage of requiring only a single online modular multiplication by the claimant. This assumes (as outlined earlier) that the exponentiation is done beforehand. However, significant computations is required by the verifier (Bob) compared to the Fiat-Shamir or GQ scheme.

3. Security assumptions

All the protocols require the assumptions that the following problems are intractable:
For a composite integer *n*:

Fiat-Shamir – extracting square roots mod *n*

Schnorr – computing discrete logs mod a prime number *p*.

Guillou-Quisquater – extracting b^{th} roots mod *n*

9. Zero-Knowledge Identification from a Geometric Perspective

In their paper, "Identification by Angle Trisection", Burmester *et al*, [13], describe an elegant zero-knowledge scheme based on the impossibility of trisecting an angle using a ruler and compass only.

1. Setup

Alice publishes a copy of an angle Y_A, which is constructed by Alice as the triple of an angle X_A, that she has constructed at random. Because trisection of an angle is impossible using a ruler and compass only, Alice is confident that she is the only one who knows X_A.

2. Protocol actions

It follows the standard form of an iterated 3-round protocol:

(ii) Alice gives Bob a copy of an angle *R*, which she has constructed as the triple of an angle *K* that she has selected at random.

(iii) Bob flips a coin, and tells Alice the result.

(iv) If Bob says "heads", Alice gives Bob a copy of the angle *K* and Bob checks that $3 * K = R$. If Bob says "tails", Alice gives Bob a copy of the angle

$$L = K + X_A,$$

and Bob checks that

$$3 * L = R + Y_A.$$

The three steps are repeated *t*-times independently. Bob accepts Alice's proof of identity only if all *t* checks are successful.

10. Converting Identification to Signature Schemes

An identification scheme involving a witness-challenge-response sequence can be converted to a signature scheme as follows:

Replace the random challenge *e* of the claimant by the one-way public hash function *h* of the witness *x* and the message *m* to be signed. This, in effect, converts an interactive identification scheme to a non-interactive signature scheme. The challenge *e* must typically be increased to avoid off-line attacks on the hash function.

11. Conclusion

In this paper we have discussed the need for identification protocols that ensure that two parties prove their identity to each other before the actual transfer of information takes place between them. We focussed our attention upon identification schemes that are based on challenge-response protocols. In particular, we examined in detail those protocols that are based upon zero-knowledge proofs where the claimant can prove their identity to another entity in real-time without revealing any meaningful information other than the claim of being that particular entity.

We touched briefly on modern day applications of these identification protocols. As the Internet grows and becomes an essential part of our lives, e-commerce has grown and with it the need for entities to identify themselves by revealing more and more sensitive information about themselves. The Internet provides a vast array of ways in which people's privacy can be and is being intruded upon, and adds new dimensions to existing problems. It necessitates the negotiation of a whole new set of balances among the various interests. Identification protocols that can prove an entity's identity without the need for that entity to reveal any information about itself are to be welcomed. As a result, we anticipate that identification protocols will grow in importance as we seek new ways to negotiate the conflicting needs of privacy on the one hand and proving our identity on the other.

12. References

1. Menezes, P, Van Oorschot, P and Vanstone, S (1997), *Handbook of Applied Cryptography*, CRC Press Inc
2. Stallings, William (1998), *Cryptography and Network Security*, Prentice Hall
3. Stinson, Douglas R, (1995), *Cryptography – Theory and Practice*, CRC Press Inc.
4. Barr, Thomas, (2002), *Invitation to Cryptography*, Prentice Hall
4. Trappe, W and Washington, L, (2001), *Introduction to Cryptography with Coding Theory*,Prentice Hall
6. http://www.crypto.nkfust.edu.tw/infosec/faq/html/14.html

7. http://www.anu.edu.au/people/Roger.Clarke/EC/AuthModel.html011019.html
8. http://www.iks-jena.de/mitarb/lutz/security/cryptfaq/q107.html
9. Quisquater, J and Guillou, L., *How to explain zero-knowledge protocols to your children*, Advances in Cryptology – CRYPTO '89
10. Lamport, L, (1981), *Password Authentication with Insecure Communication*, *Communications of the ACM*, 24 (11)
11. http://www.acm.org/sigcomm/ccr/archive/2000/july00/nyang.pdf
12. http://www.anu.edu.au/people/Roger.Clarke/EC/AuthModel.html
13. Burmester, M, Rivest, R and Shamir, A, (1997), *Geometric Cryptography: Identification by Angle Trisection*, Department of Computer Science, Israel
14. http://www.win2000mag.com/Articles/Index.cfm?ArticleID=526
15. http://developer.novell.com
16. http://www.commweb.com/article/COM20010328S0010/1
17. Grabbe, J, *Cryptography and Number Theory for Digital Cash*, http://www.aci.net/kalliste/cryptnum.htm
18. Brands, S, *Untraceable Offline Cash in Wallets with Observers*, Advances in Cryptography – Crypto '93
19. Grabbe, J, *The Mathematical Ideas Behind Digital Cash*, http://www.aci.net/kalliste/cryptnum.htm
20. Miller, J, *Digital Cash Mini-FAQ*, http://ntrg.cs.tcd.ie/mepeirce/project/mlists/minifaq.html
21. Davies, D.W, and Price, W.L., 1992, *Security for Computer Networks*, Wiley and Son
22. http://www.anu.edu.au/people/Roger.Clarke/DV/UIPP99EA.html
23. Gallian, J, (1994), **Contemporary Abstract Algebra**, Heath
24. Fraleigh, J, (1994), **A First Course in Abstract Algebra**, Addison Wesley
25. Walace, D, (1998), **Groups, Rings and Fields**, Springer
26. Schneier, B, (1996), **Applied Cryptography**, Second Edition, John Wiley & Sons