Manansala Encryption and Authentication System

A cutting edge method of foiling rip-off artists and phishers

Imagine visiting a sleek well-designed website with fantastic discounts on products that interest you.

The page allows payments by **Paypal**, a service you have used before and trust. You also notice some logos mentioning *digital signatures* and *certificates*, names that you recognize from previous net shopping.

Feeling that the site is trustworthy, you call up the payment page and notice the familiar prompt that an *encrypted* page is on the way. Everything is similar to what you've experienced before. Even the Paypal page looks the same.

However, what you don't notice is that the **URL** is not from Paypal or some similar *trusted* service. After some time waiting for your purchase to arrive, you notice instead your credit card bill showing air tickets to Mexico that you never bought!

Such a scenario is not rare. Spoofing trusted sites and *phishing* are the craze in modern Internet banditry. Credit card numbers, bank account numbers, PIN numbers and identity theft information are regularly and illegally acquired through these crafty methods.

The **Manansala Encryption and Authentication System (MEAS)** offers a powerful defense against such rip-offs that fills in the wide gaps of the current internet security and authentication protocol. The new method uses **private knowledge** known by the end users but not generally known by the hackers to encrypt and authenticate. In fact, this private knowledge is generally what they are after in the first place!

But this private knowledge, the most *sensitive* and *confidential* information is never sent over the internet.

For example, imagine being able to purchase items by entering your credit card information into a form that never actually transmits this sensitive data. Instead it uses your private knowledge to encrypt the data, but not actually in the direct encipherment.

Instead of your confidential information, a message with an authentication code consisting of a long random number is sent back. The receiver, in this case a credit card verification agency, is able to generate the same private knowledge key and decipher the message. Your identity is verified by the **authentication code** and the purchase is approved without any credit card information traveling through cyberspace.

Please direct enquiries to p.manansala@sbcglobal.net or call 915.332.4915 or 916.751.1556.

A brief overview of the public key system

The current security protocol over the internet relies on the public key system. Public keys are generated to be distributed openly. If someone wants to send an encrypted message to another person, they encrypt a session key with that person's public key.

The receiver uses a private key to decrypt the session key, which in turn is used to decrypt the entire message.

The primary method of authentication is through digital signatures and certificates. You will often read that these signatures are equivalent to hand-written signatures in identifying persons, but this is misleading.

Digital signatures at most identify only remote computers, usually servers. For example, if a person using a public library computer calls up an encrypted page, if that computer happened to have a digital certificate it would identify the computer only. The person could claim to be anyone, and the signature is meaningless in verifying that claim.

At the same time, it is easy for anyone to set up an encrypted web page. Many people have access to set up such sites on their ordinary internet accounts that they are not even aware of. Generally all it takes is a web space account with a small extra fee for encryption. The digital certificates are owned by the internet service provider rather than their clients.

Don't digital certificates verify businesses on the internet?

No, digital certificates do not necessarily offer any information on the person or entity offering an encrypted page. Anyone can set up such a page with a little internet knowledge.

So in reality, you do not know who you are sending your vital information to through an encrypted page using modern security protocols. You may know what server is involved, but in most cases that will only lead to an ISP. Fly-by-night phishing and hacking operations can set up multiple encrypted sites remotely through the internet and often paying by regular mail.

The Manansala system offers a method of assuring that the receiver at least has the ability to verify your information such as credit card number, product serial number or rmembership identification number. Of course, the hackers will not know such data as these are the targets of all their efforts! With the Manansala system all encrypted transmissions can only be decrypted with prior knowledge of this authenticating information.

The system will allow users also to set up their own personal or group security protocols based on shared private knowledge keys or through prompts sent to receiver's asking private knowledge questions for key generation.

Is encrypted data secure?

If the data is indeed encrypted, then it is likely secure if there is no breech in the related private keys.

However, some people do not realize that their “encrypted” pages actually travel unencrypted on various lengths of its journey. Most computers do not actually encrypt messages. They send them to their ISPs for encryption. They also receive information that has been decrypted at the ISP and sent unencrypted to their computer.

These messages can be accessed unencrypted by a variety of means that are commonly employed by hackers.

The reason encryption is not done at the computer level in most cases is multi-fold. The vast majority of computers do not have strong random number generators needed for cryptography. They have instead what are known as pseudo-random number generators. The real hardware generators needed to create seeds for random numbers are too expensive and difficult to maintain for most computers, cell phones and PDAs.
The **Manansala Random Number Generator** helps address these problems.

Also, the public key system as used today is very computation intensive. The mathematical problems in solving the RSA algorithm can weight down PDAs and cell phones with small memories and processing ability. Most people notice the slow response of encrypted pages despite the fact that these are churned out by souped-up servers on optimized distributed networks.

Is the public key system different than the secret key system?

Yes, but both are really secret key systems. The public key system requires that the private key be kept secret. It does not however require transmission of the private key.

Still, private keys are kept on internet servers, which can be a security risk. A disgruntled employee, contractor or other visitor to the server location may be able to access the security files. The latter can also possibly be hacked remotely.

Most internet users do not have personal private keys on their computers. They use private keys stored at their internet service provider location.

Ideally, this information should be kept at personal locations as this would greatly complicate things for hackers. Instead of simply cracking the ISP, and gaining a treasure chest of information, the hacker must now hack each individual user. Also many internet users can only be hacked by first breaking through ISP security.

The Manansala system is designed to work from the user end on web browsers, email programs, cell phones, PDAs and other terminals without public keys.

The security keys need not even be stored in the device but merely entered at the beginning of each encrypted session.

How it works

Here is how one embodiment of our patent pending invention works:

The system is designed to work from web browsers and similar programs. Our example involves a credit card transaction over the internet.

A merchant first obtains the purchaser's name, address and other relevant information. Then a **MEAS** encryption page is sent over and the browser recognizes the extension. Immediately the MEAS program checks for certain required form elements. These are displayed in a semi-standardized way that the purchaser can recognize.

Information such as credit card, number, expiration date, security/PIN code, etc. are set up on the page by the merchant as security key information and identified as such on form input boxes. The purchaser enters the information and sends it, but is first prompted to check to assure the security key information is correct.

If the purchaser oks the information, the program proceeds to use the private knowledge to generate a random number from which a session key is selected.

The session key alone enciphers the message and authentication code included as a random string of at least 48 characters. The length of the authentication code may be set by the creator of the encryption page.

After encryption on the end user computer, the message is sent without the credit card, number, expiration date, security/PIN code and other confidential information.

When the message and authentication code are properly decrypted by the manufacturer, they are assured that the sender had private knowledge of this information.

You might ask: Can't someone just intercept the authentication code and pretend to be the purchaser?

No, because a hacker would not be able to properly encrypt the code without the correct

authenticating private knowledge. With a minimum 72 byte session key, the possibilities of a hacker guessing the right key are cryptographically-speaking very unlikely. It would be far easier just to guess the purchaser's credit card information.

Without a properly encrypted page, the merchant''s credit card verifying agency decryption procedure will render a hacker message into meaningless clutter.

Once the credit card verification is complete, the verifying agency sends a confirmation to the purchaser and merchant together with a randomly-generated confirmation code to allow them to communicate together in an encrypted mode using this code. In this case the verifying agency is a trusted third-party.

Also, it is possible that MEAS only be used for confirmation of the purchase while other communication between purchaser and seller takes place through the public key encryption system.

Why shouldn't I send confidential information encrypted over the Net?

In practice, it is a good idea to minimize sending the most sensitive information. If the message is encrypted from one end of the transmission to the other, it is highly likely to be secure.

However, there are a few considerations to keep in mind.

No well-established internet security system is proved to be unbreakable. The only unbreakable system that has solid proofs is the so-called "one-time pad" encryption. Unfortunately, the one-time pad method is completely impractical for internet authentication.

While current authentication systems, including the one in MEAS appear secure they have no logical proofs to support them. Just the fact that standard encryption systems like RSA and AES have displayed unforseen vulnerabilities is telling.

Although these problems, which include timing, side channel, chosen ciphertext and other attacks have been addressed, the inventors of the encryption system did not foresee them as weaknesses.

And it is the weakest link, rather than the strongest characteristics of these systems that will leave them vulnerable. So there is no real guarantee with any security system.

By not sending the most sensitive information, one has much greater protection against unknown factors.

Also, even a trusted agency having decrypted a message might quite innocently forward it back to the other user. While in most cases, this would be harmless it is best to know how far your encrypted message travels and where.

The MEAS protocol allows you not to send the "key" information you need to protect.

While MEAS does not allow one time pad encryption on authentication, it does allow a practical form of such encipherment on every successive transmission in the session!

After authentication, the authenticating party sends a new randomly generated session key using the same encryption mode. In the case of trusted third party authentication, this would be the confirmation code.

The new session key is parsed to generate random material for the encryption pads using the Manansala Random Number Generator (MRNG). For more information on this generator, visit the following website:

> **Manansala Random Number Generator**
> https://www29.addr.com/~apu/cgi-bin/triman.pl?random

The key is parsed into a file number, starting point number and seed number.

The resulting data is fed into the MRNG and random bytes necessary to encrypt the file are generated.

The MRNG is configured to rapidly produce cryptographically-strong random numbers without need for resident hardware seed generators. If the supplied file number, starting point number and seed number are randomly-generated from another MRNG, then we have an unpredictable system.

Although the random number generation is chaotic, the parameters of each MRNG file is known. A file of size *N* for example, would produce *key size* * N^2 cryptographically-strong random numbers for each key.

The “one-time” nature of this system is not absolute, but probalistic. By using a number of independent file databases one can produce rapid and bulk random number generation.

These files, of course, must be standardized for all versions of the MEAS-powered program.

When one-time pad encryption is used, a “pad” of random bytes equaling the message slated for encryption is generated. Each random byte in order, which are known only by both users, becomes a key for each byte in the message.

Therefore, when using MEAS, one has access, after authentication, to the only proven system of encryption known!

Technical Data

Description

Private knowledge encryption and authentication with session keys selected from random strings; message cluttering encryption and one-time pad. The patent pending design also allows integration with other encryption systems, in which case it may act primarily as the private knowledge key structure allowing authentication without sending the most valued and confidential information. The system is offered for patent licensing or investment/partnership development and the following data embodies certain but not the entire range of possible applications.

Private Knowledge Key

The recommended minimum length of the private knowledge key information is 24 bytes. The minimum key is expanded to 72 bytes. Key lengths should be customizable with a certain minimum length to assure protection. A hacker's time would be much better spent at guessing credit card information than attempting to crack MEAS keys.

Byte substitution and expansion render the user's private knowledge difficult to obtain even if somehow the expanded private key were obtained. The substitution table is scrambled by the private key before substitution.

Session Key

After the generation of a cryptographically-strong random number with end user systems like the Manansala Random Number Generator, a session key is selected using a chaotic algorithm. The selection begins at a starting point determined by the values in the session key and the same values give the quantity to search for each successive value. The odd or even nature of the quantities determines whether the selection goes forward or backward in the string.

The information used to select the values comes entirely from the expanded private knowledge key, so even if the session key could somehow be found it would not give any details on the expanded private key. No change is made to the random string during selection.

The session key is the same length as the private knowledge key. For security purposes the random block should be at least key size squared and this feature also should be customizable with a minimum safe quantity.

Encipher Rounds

The message block goes to separate series of XOR rounds, one before message cluttering and one after. At least once before cluttering, and eight times after. The number is customizable after this minimum. The first round is with the original session key. The substitution table in its original form is cyclically shifted by the sum of the quantity of the session round values. The values in the session key are substituted and the table shifted for each additional round potentially up to 256 rounds.

Message Cluttering

Random strings are interspersed into the message block in a chaotic way. The purpose is to foil plain text and other types of cryptanalysis. The session key in its final substituted state performs the message cluttering.

A 100 byte string is split using the first two bytes of the session key as a percentage and the resulting two strings are placed at the beginning and end of the block.

One-time Pad

A 100 KB file can generate a pad of 10 billion sets of 24 bytes each with a 24 byte key. The chances of a pad being reused for a 24 byte key are one in 256^{24}.

One 10-megabyte MRNG files provides 2.4 X 10^{15} cryptographically-strong random numbers with a 24 byte key. That would be enough random numbers to one-time pad ten 1 gigabyte digital movies a day for about 273 years.

Authentication

An authentication code of at least 48 bytes using 256 eight-bit characters would be sent with messages in the pre-encryption phase. The code length is customizable. The proper reconstruction of the code after decryption authenticates the sender as possessing and having sent the required private knowledge. When a trusted third party authenticates the sender, an encrypted confirmation code of at least 24 bytes is sent to both parties through different encrypted channels.

Authentication would also be optional for each message block in which the last unencrypted Nb of the previous message is XORed to first unencrypted Nb of the next message. The minimum header should be around 9 bytes. So with a minimum session key of 72 bytes, the first 9 bytes would act as authentication header for 63 bytes of message consisting of the XOR result of 9 bytes respectively from the end of the last message and the beginning of the new message block. If a bad match occurs in authentication, the user is notified.

Applications

Among the initial concepts for MEAS is a browser associated program called by webpage extensions.

Such an application can be used on “top” of existing security protocols. The system sets minimum standards for form inputs and rejects the page if these standards are not met. The user is prompted for review and approval of the actual security key information used before an initial authentication message is sent. One possibility is to restrict all form inputs on the authentication page to private knowledge key inputs.

In web and html-enabled emails, the message is sent as an embedded object or attachment.

Chat, messenger, cell phone and PDA applications would use prompts for the authentication procedure. In all cases, the protocol is customizable. Prompts may be in the form of one or a series of queries.

Another option is to store private knowledge keys on file in encrypted form for automatic authentication.

Private knowledge encryption of software, CDs, DVDs, etc. is another commercial possibility. In this case, installation requires the private knowledge key of the purchaser provided at purchase. Such a method is simple to implement for e-commerce downloads, but might require burning the keys into CDs and DVDs at purchase.

Additional Mathematics

Using a minimum 72 byte session key, the 5,184 byte random block has 5184!/(256!(5184-256)! or 1.17×10^{16497} possible selection combinations.

An attempt to analyze the message cluttering sequence would require a one-in-four guess for each byte so a 72 byte session key would mean 2.23×10^{43} possibilities.

An error of one digit will break the message cluttering and XOR operations.

For production of the large random strings, a system like the Manansala Random Number Generator using multiple files of no more than about 5 KB in size each would work efficiently.

Please direct enquires to p.manansala@sbcglobal.net or call us at 916.332.4915 or 916.761.1556.

Prototype programs of the encryption phase of MEAS written in Perl script can be found at the following URL (zipped file):

http://apu.addr.com/meas.zip