Provably Unbreakable Hyper-Encryption In the Limited Access Model

Michael O. Rabin
DEAS Harvard University
Cambridge, MA 02138
Email: rabin@deas.harvard.edu

***Abstract*— Encryption is a fundamental building block for computer and communications technologies. Existing encryption methods depend for their security on unproven assumptions. We propose a new model, the Limited Access model for enabling a simple and practical provably unbreakable encryption scheme. A voluntary network of tens of thousands of computers each maintain and update random pages, and act as Page Server Nodes. A Sender and Receiver share a random key K. They use K to randomly select the same PSNs and download the same random pages. These are employed in groups of say 30 pages to extract One Time Pads common to S and R. Under reasonable assumptions of an Adversary's inability to monitor all PSNs, and easy ways for S and R to evade monitoring while downloading pages, Hyper Encryption is clearly unbreakable. The system has been completely implemented.**

I. Introduction

A. Background

Modern encryption methods depend for their security on assumptions concerning the intractability of various computational problems such as the factorization of large integers into prime factors or the computation of the discrete log function in large finite groups. Even if true, there are currently no methods for proving such assumptions. At the same time, even if these problems will be shown to be of super-polynomial complexity, there is steady progress in the development of practical algorithms for the solution of progressively larger instances of the problems in question. Thus there is no firm reason to believe that any of the encryptions in actual use is now safe, or an indication as to how long it will remain so. Furthermore, if and when the current intensive work on Quantum Computing will produce actual quantum computers, then the above encryptions will succumb to these machines.

At present there are three major proposals for producing provably unbreakable encryption methods. Quantum Cryptography [1] employs properties of quantum mechanics to enable a Sender and Receiver to create common One Time Pads (OTPs) which are secret against any Adversary. The considerable research and development work as well as the funding invested in this effort are testimony to the need felt for an absolutely safe encryption technology. At present Quantum Cryptography systems are limited in range to a few tens of miles, are sensitive to noise or disturbance of the transmission medium, and require rather expensive special equipment.

The Limited Storage Model was proposed by U. Maurer [2]. It postulates a public intensive source of random bits. An example would be a satellite or a system of satellites containing a Physical Random Number Generator (PRNG) beaming down a stream α of random numbers, say at the rate of 100GB/sec. S and R use a small shared key K to extract a relatively small number of common bits from the stream and use those bits and the key to form OTPs which are subsequently employed in the usual manner to encrypt messages. The Limited Storage Model further postulates that for any Adversary or group of Adversaries it is technically or financially infeasible to store more than a fraction, say half, as many bits as there are in α. It was proved in [3], [4], [5] and [6] that under the Limited Storage Model assumptions, one can construct schemes producing OTPs which are essentially random even for a computationally unbounded (but storage limited) Adversary. The critique of the Limited Storage Model is three-fold. It requires a system of satellites, or other distribution methods, which are very expensive. The above rate of transmission for satellites is right now outside the available capabilities. More fundamentally, with the rapid decline of cost of storage it is not clear that storage is a limiting factor. For example, at a cost of \$1 per GB, storing the above mentioned stream of bytes will cost about \$3 Billion per year. And the cost of storage seems to go down very rapidly.

B. The Limited Access Model

The Limited Access Model postulates a system comprising a multitude of sources of random bytes available to the Sender and Receiver. Each of these sources serves as a Page Server Node (PSN) and has a supply of random pages. Sender and Receiver initially have a shared key K. Using K, Sender and Receiver asynchronously in time access the same PSNs and download the same random pages. The Limited Access assumption is that an Adversary cannot monitor or compromise more than a fraction of the PSNs while the Sender or Receiver download pages. After downloading sufficiently many pages, S and are make sure that they have the same pages by employing a Page Reconciliation Protocol. They now employ the common random pages according to a common scheme in groups of, say, 30 pages to extract an OTP from each group. Let us assume that the extraction method is simply taking the XOR of these pages. The common OTPs are used for encryption in the usual manner.

A crucially important point is that a Page Server Node sends out a requested random page *at most twice*, then destroys and replaces it by a new page. Opportunity knocks only twice!

Why is this scheme absolutely secure? Assume that we have 25,000 voluntary participants acting as PSNs. Assume that a, possibly distributed, Adversary can eavesdrop, monitor or corrupt (including by acting as imposter) no more than 5000 of these nodes. Thus the probability that in the random choice of the 30 PSNs from which a group of 30 pages are downloaded and XORed, all 30 pages will be known to the Adversary is smaller than $(1/5)^{30}$, i.e., totally negligible. But if an Adversary misses even one page out of the 30 random pages that are XORed into an OTP then the OTP is completely random for him.

The "send at most twice, then destroy" policy prevents a powerful Adversary from asking for a large number of pages from each of the PSNs and thereby gain copies of pages common to S and R. The worst that can happen is that, say, S will down load a page P from PSN_i and the Adversary (or another user of Hyper-Encryption) has or will download the same page P from PSN_i. When R now requests according to the key K the same page from PSN_i, he will not get it. So R and S never have a page P in common if P was also downloaded by a third party. The only consequence of an Adversary's downloading from too many PSNs is denial of service to the legitimate users of the system. This is a problem for any server system and there are ways of dealing with this type of attack.

What if an Adversary eavesdrops onto the Sender and or Receiver while they are downloading pages from PSNs. Well, S and R can go to an Internet café or one of those establishments allowing a customer to obtain an Internet connection. They can use a device that does not identify them and download thousands of pages from PSNs within a short time. The salient point is that S and R need not, in fact preferably should not, time-synchronize their access to the PSNs. Once S and R have common OTPs, they can securely communicate from their fixed known locations with immunity against eavesdropping or code breaking.

The initial key K is continually extended and updated by S and R using common One Time Pads. Each pair of random words from K is used to select a PSN and a page from that PSN *only once* and then discarded. This is essential for the absolute security of Hyper Encryption.

With all these provisions Hyper Encryption in the Limited Access Model also provides Ever Lasting Secrecy. Let us make a worst case assumption that the initial common key K or its later extensions were lost or stolen after their use to collect common random pages from PSNs. Those pages are not available any more as a result of the send only twice and destroy policy. Thus the extracted OTPs used to encrypt messages cannot be reconstructed and the encryption is valid in perpetuity. By contrast, all the existing public or private key encryption methods are vulnerable to the retroactive decryption attack if the key is lost or algorithms come up that break the encryption algorithm.

In Section III we shall describe an additional scheme based on the use of search engines for the generation of OTPs and of unbreakable encryption.

Our systems were fully coded in Java for distribution as freeware amongst interested users. All the protocols described below are running in the background on the participating computers and impose negligible computational and storage overheads on the host computer.

II. THE PAGE SERVER SYSTEM AND ITS PROTOCOLS

A. The Page Server Node and its Protocols

Every PSN has a name and an IP. We shall refer to the PSN named j as PSN_j and to its IP address as $IP(j)$. It has a physical generator of random numbers $PRNG_j$. In one implementation we use a cheap radio plugged into the microphone port. The radio produces random white noise which is sampled a few tens of thousands of times a second. The lowest binary digit of the amplitude of a sample point is fairly random. To enhance randomness we XOR those bits in groups of six. In this way we produce a stream of random bits at a rate of about 10K/sec.

The random bits are assembled into pages of 4000 bytes each. At any given time PSN_j maintains a list of about a thousand random pages $P_1, \cdots, P_{1000}$. It also maintains a list of 32-bit words $w_1, w_2, \cdots, w_k$ representing past uncompleted requests for pages. A Hyper Encryption user U requests a page from PSN_j by sending in a 32-bit word u.

If $u = w_m$ for some $1 \leq m \leq k$, then PSN_j sends to U the page P_m, removes P_m from the list, wipes P_m from memory and replaces this page by a new random page appended at the end of the list of pages. The word w_m is also removed from the list of words. If u does not appear in the current list of words then PSN_j appends $w_{k+1} = u$ at the end of the word-list and sends P_{k+1} to U (in this order!).

These procedures enforce the "send a page at most twice" regime. The PSN also scrubs the lists from a page requested once if that page was not requested again within a set time.

B. The User Device and its Protocols

Let U be a user device that is one of a pair of partners Alice and Bob who employ Hyper Encryption. Alice and Bob share a key $K = u_1, u_2, \cdots, u_{2s-1}, u_{2s}, \cdots$, of randomly chosen 32-bit words. They also share a list of the current PSNs, $PSN_1, \cdots, PSN_j, \cdots$. If available, they also store the IPs *IP(j)* of those PSNs.

B1. Requesting and Downloading Random Pages

When U, which is either Alice or Bob, prepares to download a page to be locally called PU_s it uses the pair u_{2s-1}, u_{2s} out of K. There is a mapping F so that $F(u_{2s-1}) = j$ is a name of a server PSN_j. U sends u_{2s} to PSN_j , gets back a page P_m and sets $PU_s = P_m$. If PSN_j is not responding , U will use other pairs of words and re-use the pair u_{2s-1}, u_{2s} a specified number of times within a specified time interval until success or discarding of the pair.

B2. Page Reconciliation Protocol

The "send a page at most twice" regime, and the asynchrony of the download procedures of Alice and Bob, imply that for a given s and at a given time, $PA_s \neq PB_s$ or $PA_s = nil$, are possible. We thus need a Page Reconciliation Protocol.

From time to time Alice scans the list of downloaded and not yet used pages $PA_{s'}, PA_{s''}, \cdots$. Using a common hash function H, Alice computes and sends the list of pairs $(s', H(PA_{s'})), (s'', H(PA_{s''})), \cdots$, to Bob. Bob scans his list of downloaded and not yet used pages $PB_{r'}, PB_{r''}, \cdots$. Computing hash values of these pages and comparing, Bob prepares a list of indexes $t', t'', \cdots$, of pages common to Alice's and his list of not yet used pages and sends these indexes to Alice.

The pairs $(t', PA_{t'}), (t'', PA_{t''}), \cdots$, of verified common random pages are appended, in ascending order of the indexes, by Alice and by Bob to their lists of verified common pages. The above process is also started from time to time, independently of Alice, by Bob according to some internal schedule.

B3. Secrecy of the Exchange of Page Reconciliation Messages

The use of the hash function H is not intended to hide information about the pages to be reconciled. Absolute secrecy for these pages is assured by Alice and Bob employing an existing common OTP to Hyper Encrypt their page reconciliation messages. To enable getting off the ground, the initial shared key K contains sufficiently many random bytes some of which are used as OTPS for the first few page reconciliation messages.

B4. Forming One Time Pads

According to the above, at any given time after the first page reconciliation round, Alice and Bob have a common ordered list of common random pages. From time to time Alice and Bob, independently of each other, remove pages in groups of 30 starting at the beginning of the list. They form an OTP from each group by byte-wise XORing the pages within the group. Thus for pages $P_1, \cdots, P_{30}$:

$$OTP = P_1 \oplus P_2 \oplus \cdots \oplus P_{30}.$$

The thus obtained One Time Pads are distributed according to a fixed rule, common to Alice and Bob, into four groups. The bytes from Pads within each group are appended in a prescribed manner to one of four corresponding arrays of random bytes: (1) AtoB[ ] ; (2) BtoA[ ] ; (3) SYSTEM[ ] (4) KEY[ ].

B5. The Key K

We note that according to B1 and B4 the common key K is actually an array KEY[ ] treated as a FIFO queue from which bytes are removed from the beginning in groups of eight bytes treated as pairs of 32-bit words. Additional data needs to be incorporated to serve as the identifier s of a pair of words used to download the page referred to in B1 as PU_s. The salient point is that every pair of words from K is used only once.

B6. Sending and Receiving Encrypted Messages

Every participating user device has a Hyper-Encrypted Mail utility, also supplied in Java code in our system. When Alice wishes to send an encrypted message M comprising m bytes to Bob, she selects an interval $[k, m+k-1]$ and sends to Bob the byte-wise XORed message $C = M \oplus$ AtoB $[k, m+k-1]$ as well as the relevant interval $[k, m+k-1]$. The bytes AtoB $[k, m+k-1]$ used to encrypt M are removed by Alice from AtoB[ ] after Bob acknowledges receipt of C.

Upon receipt of C and $[k, m+k-1]$, Bob decrypts by forming $C \oplus$ AtoB $[k, m+k-1] = M$ and acknowledges receipt to Alice. He also removes from AtoB[ ] the bytes AtoB $[k, m+k-1]$ used in the encryption/decryption.

The above procedure, augmented by careful book-keeping steps to ensure that the array AtoB[ ] always remains synchronized between Alice and Bob, ensures that any sequence of random bytes employed as an OTP for encryption is used only once.

The use-only-once requirement is also the reason for having two arrays AtoB[ ] and BtoA[ ]. If Alice and Bob were to employ the same array of random bytes then a situation could arise where Alice sends to Bob a message M and Bob sends to Alice a message N, and the intervals of random bytes used as OTPs by Alice to encrypt M and by Bob to encrypt N are overlapping. This creates a breach in security. If an OTP is used to encrypt two different messages, partial information on the messages can be derived.

The Hyper-Encrypted Mail utility is also employed automatically in the above procedure $B3$ to maintain secrecy in the exchange of page reconciliation messages.

B7. Message and Sender Authentication

Hyper Encryption can be augmented by absolute authentication procedures. An efficient way for doing this employs fingerprinting by an irreducible polynomial [7]. We require a fifth common random array AUTH[ ] for Alice and Bob, or else we can employ bytes from the arrays they use as OTPs, being careful to do proper book-keeping of consumed bytes to avoid ambiguities.

Let us employ irreducible polynomials of degree 31. According to [7], since 31 is a prime, once we have a single irreducible polynomial of degree 31, $f(x) \in F_2[x]$, we can efficiently compute random irreducible polynomials $g(x)$ of degree 31 from random words w in {0, 1}. I.e. there is an efficiently computable function $I(w) = g_w(x)$ so that if $w \in \{0,1\}^{32}$ (and w unequal 00000001 or 00000000) then $g_w(x)$ is irreducible of degree 31.

To authenticate a message M, Alice pads M on the left with 31 0's. Assuming M to be already padded, $M = a_0 a_1 \cdots a_m \in \{0,1\}^{m+1}$. We consider M as a polynomial in $F_2[x]$. Note that we are viewing M as a sequence of bits, not bytes. This is done just for convenience of the presentation. Thus,

$$F_M(x) = a_0 + a_1 x + \cdots + a_m x^m$$

Alice now selects the four bytes AUTH$[r, r+3]$. Disregarding the last bit, she obtains a 31-bit word w. If w is not one of the above two exceptions, Alice computes the irreducible polynomial $I(w) = g_w(x)$. The fingerprint of M with respect to $g_w(x)$ is the residue of $F_M(x)$ when divided by $g_w(x)$:

$$FING(M, g_w(x)) = F_M(x) \; mod \; g_w(x).$$

Alice forms $M_1 = M^{*}FING(M, g_w(x))$, where $*$ denotes concatenation. She Hyper Encrypts M_1 as in B6 and sends the cipher text C_1 with the indexes defining the OTP used, and with the above r to Bob.

Bob decrypts C_1 as in B6. To authenticate M, he recovers $w =$ AUTH$[r, r+3]$ (discarding the last bit). Bob computes $I(w) = g_w(x)$ and $FING(M, g_w(x)) = G$. He now verifies that the decrypted C_1 is indeed $M * G$.

It can be proved that an Adversary cannot produce any cipher text C_2 different from one sent by Alice, which will pass this test with probability $\geq m/2^{31}$. Now, this probability is smaller then $m10^{-9}$, where m is the length of the message M. Thus both the message and the Sender Alice are robustly authenticated. For the faint of heart: The use of two random polynomials of degree 31 will reduce the probability of cheating below $m^2 10^{-18}$.

It is worthwhile noting that the production and listing of the random irreducible polynomials $g_w(x)$ of degree 31 can be done by Alice and Bob off line, ahead of encrypting or decrypting messages. This reduces authentication of messages at send/receive time to computations of fingerprints using a given $g_w(x)$. This is a very rapid computation involving only one XOR operation per message bit.

III. SEARCH ENGINE BASED HYPER ENCRYPTION

Search engines can serve as another source of random OTPs in the Limited Access model for Hyper Encryption. We shall give a brief outline of a search engine based system. Consider a typical search engine, to be called Pimpernel.

Alice and Bob share a random key $K = w_1, w_2, \cdots$, from which they consume a few bytes at a time to download a random page from a random server on the Web.

Asynchronously in time, they employ w_1, w_2, w_3 to look up a common ordinary dictionary and obtain an English word or phrase such as "blue sky".

Alice's (and similarly Bob's) computer U Pimpernels "blue sky" and gets a list of web pages. Using w_4, U randomly selects and downloads one of the first 20 pages. Using w_5, U randomly selects one of the links of the downloaded page and downloads the page referred to in the link. This process is repeated to depth six.

So now U has a web page P. This page in not a string of random bytes. There are well studied randomness extractors for producing a highly random string of bytes from a weakly random source [8]. As a practical matter we first compress P and then encrypt using a private key common to Alice and Bob. The resulting page, call it P_1 (because w_1 was used to initiate its production) is appended by U to the list of random downloaded pages.

In all other respects the search engine based system is configured like the page server system of Section II.

In our implementation we combine pages downloaded from PSNs with pages obtained through the use of a search engine.

The search engine based Hyper Encryption is also implemented at Georgia Tech. by Yan Zong Ding.

IV. CONCLUSIONS

We have introduced the Limited Access Model as a basis for provably unbreakable encryption. We described two methods within the Limited Access Model for a Sender and Receiver to easily and practically create a stream of common random One Time Pads to be used for encryption and authentication. Under reasonable practical assumptions the PSN based implementation is directly shown to be provably unbreakable. The theoretical analysis of the search engine based system, when not used in combination with the PSN system is not that clear. But we believe that in practice the latter system by itself also provides very high security.

Other sources of randomness within the Web can be employed for Hyper Encryption in a manner similar to that of the search engine based system.

ACKNOWLEDGMENT

The following Harvard students have brilliantly worked on the architecture and implementation of the systems described in this paper. Y.C. Chang, M. Hamburg, C. Martin, B. Parno, A. Rampell, M. Scnall-Levin, L. Storzek, K. Venkatram, D. Xiao.

This research was supported in part by National Science Foundation grant CNS-0205423 at Harvard University.

REFERENCES

[1] C.H. Bennett and G. Brassard, "Quantum cryptography: Public key distribution and coin tossing,"Proceedings of IEEE International Conference on Computers, Systems and Signal Processing, Bangalore, December 1984, pp. 175-179.

[2] U. Maurer, "Conditionally-perfect secrecy and a provably-secure randomized ciper," *Journal of Cryptology,* 5 1992, pp. 53-66.

[3] Y. Aumann and M.O. Rabin, "Information theoretically secure communication in the limited storage model," Advances in *Cryptology-Crypto 99,* Lectures in Computer Science 1666, Springer Verlag, 1999, pp. 65-79.

[4] Y. Z. Ding and M.O. Rabin, "Hyper-encryption and everlasting security," *Proc. of the 19th International Symp. on Theoretical Aspects of Computer Science (STACS),* Springer-Verlag LNCS 2285, 2002, pp. 1-26.

[5] Y. Aumann, Y. Z. Ding and M.O. Rabin, " Everlasting security in the bounded storage model"*IEEE Transactions on Information Theory,* vol. 48, 6, 2002, pp. 1668-1680.

[6] S. Dziembowski and U. Maurer, "Optimal randomizer efficiency in the bounded-storage model," *Jour. of Cryptology,* vol. 17, 1, 2004, pp. 5-26.

[7] M.O. Rabin, "Finger printing by random polynomials," Computer Science, DEAS, Harvard University, TR-15-81, 1981.

[8] R. Shaltiel, "Recent developments in extractors (Survey paper)," Current trends in theoretical computer science. The Challenge of the New Century, vol 1: Algorithms and Complexity.