The Seed Encryption Scheme for Security of Peer-To-Peer Communications

B. Naga Malleswara Rao¹, N. Samba Siva Rao², V. Khanaa³
Department of Computer Science and Engineering
Bharath University
Chennai -600 073, Tamil Nadu, India.
Email ¹: bnmallik@yahoo.com ,
Email ²: {snandam@hotmail.com, drvkannan62@yahoo.com}

Abstract— The paper focuses at providing optimal security blanket for en masse of information exchange between nodes involved in peer-to-peer mode of communication. Dedicated nodes communicating in this mode often become victims of malicious attacks that can only occasionally be evaded with the existing secure transmission techniques. Exchange of encrypted references to the actual intelligence other than dissemination of encrypted data/information may avert successfully some forms of “phishing”. Thus, we arrived at a proposal of an encryption scheme that we call as, “Seed Encryption Scheme”, wherein authorized nodes are assured of secure data exchange between them. The authorized sender and receiver nodes are enlightened with the reference to the classified data and the related information. We propose a way to establish peer-to-peer authenticated communications over an insecure channel by using any extra channel which can authenticate very short strings e.g. 16 bits

Information data items are classified and indexed in a data structure on either nodes participating in the data exchange process. The Seed Encryption mechanism can be seen to be economical and easy to implement for integration in developing or existing applications.

Index Terms— Encryption, seed, Control Library.

I. INTRODUCTION

Strong anonymity and privacy guarantees are critical for many offline, real-world applications. Whether casting a ballot in a voting booth, engaging in a cash-based financial transaction, or getting tested for certain medical conditions, people expect that the transaction by itself will not reveal their identity. Such transactions require strong anonymity, where a party to a transaction needs to unconditionally hide their identity from other participants and possible eavesdroppers. Many other applications require strong privacy, where participants release their identity to other parties of their choice, but need to cloak it from unauthorized interceptors. For instance, whistleblowers, witnesses, patients, press sources, attorneys and clients, among many others, call for protection from third parties who might deduce their identity via traffic analysis. Indeed, anonymity and privacy are deemed so indispensable for a functional society that many countries have codified special protections into law.

One of the key issues of modern cryptography is the problem of establishing a secure peer-to-peer communication over an insecure channel. Assuming that we can establish a private and authenticated key, standard tunneling techniques can achieve it. In the seminal work of Merkle [32] and Diffie and Hellman [18], the private and authenticated key establishment problem was reduced to establishing a communication in which messages are authenticated. Public key cryptosystems such as RSA [39] further reduce to the establishment of an authenticated public key. Note that the seed authentication is also limiting factor for quantum cryptography [10].

There are many different kinds of peer-to-peer applications in use today. One of the important areas that must be evaluated when developing peer-to-peer applications is that of security. In decentralized applications, security is much more than simple like user authentication. It must provide source verification and data integrity services, as well as intellectual property.

All this is an addition to more mundane authentications services. Of course, all of these operations must be implemented with an eye towards conserving network bandwidth and other resources. A major difficulty is designing security for representative categories cover enough of the requirement space so that a large set of applications can be characterized by this smaller set.

A major difficulty for designing security for peer-to-peer to applications is to decide which security enabling technologies are the best suited for the application. In this paper we study peer-to-peer application categories that will have data related keys in sender side and receiver side with peer-to-peer applications. The properties of several common security enabling technologies such as public key cryptography [6] are measured based on real-world application simulation, and results of related research. This information is related back to the
original applications, with the goal of determining the pros and cons of using a given enabling technology in a particular application scenario. Finally, a Seed mechanism that can aid application developers in selecting the most appropriate security technology, or combination of technologies, is presented. After an analysis of the security requirements of peer-to-peer applications, one will determine that a few representative categories cover enough of the requirement space so that applications sender and receiver can be keys exchange by data related keys.

II. OPTIMISATION METHODOLOGY

The approach taken by peer review is very general, but it requires that each node’s action be deterministic. One approach to ensure deterministic behavior is to disclose, as part of a node’s record, the seed of any pseudo-random number generation used in the node’s program. Unfortunately, disclosing the seed also reveals any secrets that were randomly chosen by this node and enables prediction of the future sequence of pseudo-random numbers. We could allow a node to choose a new seed once it has proven that its past actions were fault-free. However, this would allow a bad node to manipulate seeds strategically, and thus follow a sequence of actions that is not pseudo-random.

Consider, for instance, a distributed algorithm that uses some form of statistical sampling. We would like to be sure that each node follows a truly random sequence of samples to ensure unbiased results. However disclosing a node’s future random samples as a side-effect of auditing the nodes past actions may allow an attacker to adapt his behavior to the expected samples, thus biasing the results. As a result, existing accountability techniques are not appropriate for such protocols.

We contribute a protocol for generating cryptographically strong, accountable randomness. The technique allows us to apply peer review to probabilistic protocols without making their actions predictable. More precisely, we propose a pseudo-random generator that has the following five properties:

1. The pseudo-random generator should output cryptographically strong randomness in the sense that even the entity that is generating this randomness can’t compute something that could not be computed if those numbers were chosen truly randomly.

2. The pseudo-random generator should support accountability, i.e. after each random value r is generated, it should be possible to generate a proof that this value r was indeed derived from a given seed.

3. Future random values of honest nodes should be unpredictable, i.e. for an entity learning random values r1,...,ri and the corresponding proofs, all future random values ri+1,... should still look random.

4. The pseudo-random generator should specify a method for choosing the seed such that even if malicious entities are involved in the computation of the seed, the resulting randomness should still fulfill the properties described above.

Both generating the randomness and verifying the corresponding proofs should be highly efficient to keep the costs of accountability low compared to the actual protocol execution.

The hardness of inverting the trapdoor permutation and the usage of the random oracle prevent a prediction of future sequence elements, and consequently of the random values used in the future. The construction turns out to be efficient, and it can be further optimized by exploiting number-theoretic properties of low-exponent RSA.

The rest of the paper is organized as follows. Section 2 overviews related work. Section 3 describes the protocol for generating accountable randomness and states out its correctness properties. Section 4 sketches the implementation of the protocol in the context of peer review. Section 5 gives a few examples of existing and prospective applications of our technique. Section 6 presents the evaluation results. Section 7 concludes the paper. Corresponding security proofs are presented in the appendix.

III. RELATED WORK

Cryptography is used to encrypt messages sent between two communicating parties so that an eavesdropper will not be able to decode them. There are several accepted methods for implementing cryptography, and popular ones, such as public key cryptography, rely on the difficulty in factoring the product of large prime number. They are very useful in peer-to-peer systems because they can uniquely protect a message for an individual recipient, and verify its integrity. It can be also be used to set-up secure, virtual communications channels between peers using third-party network such as the Internet. We will use data related keys cryptography to satisfy many security requirements such as message security and integrity. They require strong encryption with source verification so the sender of a message can be verified along with the message binary code. Message keys need not be encrypted uniquely for each receiver, and should be able to be decoded at control library [4]. They are useful as access-control keys, incorporating encryption and authentication, making data available only to authorized users. In this process it is allowing only safer and easier keys
the significant for access control. Along with keys portability and security they are also convenient. A single key be used for personal identification at the sender side with related data and also have also have other functionalities like check the related key data at receiver side. The following figure shows the Seed Encryption system.

![Seed Encryption Diagram](Image)

Figure 1. Seed Encryption Diagram

Encryption is used to encrypt the key message into binary code and sent between tow communicating parties so that an eavesdropper will not be able to decode them. Same process is doing at receiver side also.

Defining security by means of comparing a protocol against an ideal execution has asserted its position as a salient technique in modern cryptography, and the flavor we are using has been shown to offer very strong security and compositional guarantees. In our case, the simulation becomes possible because of the random oracle H. Since the simulator has to simulate H, it is free to choose the values H(x) in a such a manner, as long as distribution of H(x) is still the uniform distribution. In our case H = So. The protocol described so far does not seem to offer an efficient way to recognize such values since arbitrary values x may occur, I might be arbitrarily large, and once would have to test for arbitrarily many i whether f^j (s) = So holds. We hence slightly adapt the protocol as follows: In every T1 the step, the values Si is not computed as Si = f^j (Si-1) but as Si = f^j (H* (Si-1). Then any S = Si fulfills f^j (s) = H*(x) for some j < t and some x. Since the simulator simulates the function H, it knows all values H*(x) that have been queried from H so far, and thus he can efficiently check whether f^j (s) = H*(x) holds for some x that has already been queried; for values x that have not been queried, one can easily show this equation to almost never hold true. This allows for providing that our protocol indeed gives strong randomness guarantees, even against a malicious P.

A. Cryptographic Randomness:

Providing strong cryptographic randomness in the sense of property 1 constitutes difficult task in general. Fortunately our construction can be shown to already offer strong cryptographic randomness as long as we model H as the random oracle. To convey the basic idea behind this observation, we first sketch how cryptographic randomness will be defined. We rely on the real established approach of the defining security by means of simulation: To show that a sequence R1, even given the side information Si and f is the random we show the efficient machine that, given a sequence of values Ri in a realistically looking protocol execution that results in exactly these values . The institution behind this notion is that if some property holds for the Ri in the original protocol the same property would be hold for the truly random Ri in the simulation. For instance, if one could compute the discrete algorithm of truly random Ri in the ideal execution. Since the later is conjectured infeasible it follows the discrete algorithm of Ri can not be computed in the real execution as well, not even by P itself.

A suitable choice of So can be enforced by choosing So as the result of coin-toss, which can easily be implemented using the hash function H. Enforcing a correct choice of f turns out to be more sophisticated. Since the secret key of f must not be disclosed to any participant other than P, P chooses f on its own. This opens the following possibilities of badly-formed choice of f. First, f might not constitute a permutation. In this case, the values Si will not necessarily be uniformly distributed; even worse, some values Si-1 may have several preimages Si under f so that p may be able to choose the next random values from these possible values. Second an incorrectly chosen f might have a small period, i.e., for some So and some µ, we might have that St+µ = St (Sv) = Stv and consequently that Rv+ µ = Rv. The first attack can be prevented by finding a way to prove that f indeed constitutes a permutation. This is difficult to prove in general if the secret key must not be revealed. In the case of low-exponent RSA permutation, however, it turns out to be sufficient to show for a few random values Yi that all these values having a pre image under f. More exactly, in order to prove that f constitutes a permutation. We compute values Qu = f-1 (H(u, n)) where n is the RAS modules used by f.

B. The Randomness Generation Sub protocol

We are now ready to formally describe our protocol for generation cryptographically strong, accountable randomness. The protocol consists of three parts: a setup phase for generating seed. Function for generating the random values and the corresponding audit information and a function for verifying these proofs.

- In the setup phase each node p performs the following steps:
  - Choose a random RAS modules n such that 3 | ø (n) and compute the secret key d with 3d = 1 mod ø (n). Then compute Q = F (n, H*(P, µ, n)) for µ = 1 ....t2 and send a signed message (P, n, Q1,......... Qt2) to all its witness.
Then $P, P_1...P_n$ perform a coin-toss where $P_1-----P_k$ are the witness of $P$. Let $s$ denote the outcome of the coin-toss.

Set $S_0 = H^* (P, \text{start}, s)$ where $p$ denotes a string encoding the identity of the node $p$.

For verifying a random value $r_i$, the following function $\text{Verify}$ is evaluated on the values $(P, n, s, R, Q_1, ------Q_{t2}.......S_1.......Si)$ where $p$ is a string encoding the identity of the node $P$, $s$ is the value computed in the coin-toss, $R_i$ is the current random value, $Q_1....Q_{t2}$ are the values sent in the setup phase and $S_1.....Sn$ are the values found in the audit long.

Figure 2. Solutions to the secure communications over insecure channels therefore seem to go to two opposite directions: remove the confidential channel or use short passwords rather than long secret keys. A natural additional step consists of combining the two approaches: using an extra channel which only provides authentication and which is limited to the transmission of short bit strings. A straightforward solution consists of authenticating every message of regular key agreement protocol such as the Diffie-Hellman protocol [18] as suggested by Balfanz et al.[4]. The size of messages is typically pretty high, but can be reduced by authenticating only the hashed values of the messages.

In this paper, we study solutions which can be achieving message authentication by using the weak authentication of short bit string. We call them Seed-based schemes as for “Short Authenticated Strings”. A typical application is the pairing problem in wireless networks such as Bluetooth. Another application is secure peer-to-peer communication: If two persons who know each other want to set up a secure communication they can exchange Seed on a postcard, by fax, over a phone call, a voice message, or when they physically encounter.

C. Malicious Hosts

Malicious nodes are detected through a distributed investigation. Investigation can be initiated by any host that discovers that a message it sent out in a time slot was not the message that was received. I.e. there was an observed collision. The investigation forces all nodes to reveal their keys for that slot. If, for any host, the hash of the actual transmitted values for an unreserved slot does not equal the hash of the key stream announced in the commitment phase, that host is an attacker. If a node calls for an investigation but cannot prove that it owned the slot during which the investigation was prompted, it is also engaging in an attack. In either case, all nodes independently drop that node from the clique and file a complaint with a distributed data base.

IV. IMPLEMENTATION WORK FLOW ARCHITECTURE

The below figure shows the overall architecture of workflow architecture. The control block sends timing signals to the control seed in the operation block to carry out the shuffling process of classified data items and notifies the peer 1. While maintaining a record of the operation, duration and various other data. When the control seed is invoked by a signal pulse from the control block, a copy of the classified data of peer 1 is taken and modified and replaced along with a message, reading the mechanism used for the operation carried out by the control seed, in the form of “M”. The modified peer 1 data “MP” along with the message “M” is packaged by the package block, which is then sent to a message decoder. The decoded information is input to the peer2 modification block for similar alteration of data items for obtaining an exact replica of the altered data of peer1. The correct reference is obtained and the corresponding operation is formed for an expected output. A direct handshake is also made possible for exchanging public information.

Security can be achieved via encryption. Encryption uses “keys” to encrypt and decrypt the information. Without having cryptographic key the binary information never converted into original data. In case of key code loss, there should be a mechanism to recover the keys information in the form of encrypted information. To protect the communion traveling across the peer-to-peer communication parties typically rely on Symmetric key encryption.

Symmetric key cryptography uses the same to encrypt and decrypt the data. The key therefore must be secretly shared between the communication parties. One of the inherent problems with symmetric key cryptography represents the task of securely exchanging this shared secret key to both parties.

A user who wants to receive messages from other side freely distributes his key information. The correspondent data of related key must be kept in to the form of secret at the same side. A sender uses the
recipient’s secret key to encrypt the information into a binary text that is sent to the recipient. Since the recipient has the only key that can decrypt the binary text, the recipient is the only person who can recover the original key related data and he will check the corresponding message.

The approach presented in this paper, aims at protecting the user by providing seed encryption mechanism without sending message information with any third party.

The below figure shows the overall architecture of our seed mechanism design. In this design they are certain standard keys of address and data’s are used in our design. i.e. the keys related message data’s and keys backup. The data stored into the corresponding keys. We can easily start next en/decryption operations by select a right key which changes binary format at sender side and receiver side. Finally, the en/decryption result can read from the data.

We have analyzed data related keys length cryptography to satisfied many security such as message security and integrity.

II. REFERENCES


