J. Cryptology (1988) 1: 65–75

Journal of Cryptology

The Dining Cryptographers Problem: *Unconditional Sender and Recipient Untraceability*

David Chaum
Centre for Mathematics and Computer Science, Kruislan 413, 1098 SJ Amsterdam, The Netherlands

Abstract. Keeping confidential who sends which messages, in a world where any physical transmission can be traced to its origin, seems impossible. The solution presented here is unconditionally or cryptographically secure, depending on whether it is based on one-time-use keys or on public keys, respectively. It can be adapted to address efficiently a wide variety of practical considerations.

Key words. Untraceability, Unconditional Security, Pseudonymity.

Introduction

Three cryptographers are sitting down to dinner at their favorite three-star restaurant. Their waiter informs them that arrangements have been made with the maître d'hôtel for the bill to be paid anonymously. One of the cryptographers might be paying for the dinner, or it might have been NSA (U.S. National Security Agency). The three cryptographers respect each other's right to make an anonymous payment, but they wonder if NSA is paying. They resolve their uncertainty fairly by carrying out the following protocol:

Each cryptographer flips an unbiased coin behind his menu, between him and the cryptographer on his right, so that only the two of them can see the outcome. Each cryptographer then states aloud whether the two coins he can see—the one he flipped and the one his left-hand neighbor flipped—fell on the same side or on different sides. If one of the cryptographers is the payer, he states the opposite of what he sees. An odd number of differences uttered at the table indicates that a cryptographer is paying; an even number indicates that NSA is paying (assuming that the dinner was paid for only once). Yet if a cryptographer is paying, neither of the other two learns anything from the utterances about which cryptographer it is.

To see why the protocol is unconditionally secure if carried out faithfully, consider the dilemma of a cryptographer who is not the payer and wishes to find out which cryptographer is. (If NSA pays, there is no anonymity problem.) There are two cases. In case (1) the two coins he sees are the same, one of the other cryptographers said "different," and the other one said "same." If the hidden outcome was the same as the two outcomes he sees, the cryptographer who said "different" is the payer; if the outcome was different, the one who said "same" is the payer. But since the hidden coin is fair, both possibilities are equally likely. In case (2) the coins he sees are

different; if both other cryptographers said "different," then the payer is closest to the coin that is the same as the hidden coin; if both said "same," then the payer is closest to the coin that differs from the hidden coin. Thus, in each subcase, a nonpaying cryptographer learns nothing about which of the other two is paying.

The cryptographers become intrigued with the ability to make messages public untraceably. They devise a way to do this at the table for a statement of arbitrary length: the basic protocol is repeated over and over; when one cryptographer wishes to make a message public, he merely begins inverting his statements in those rounds corresponding to 1's in a binary coded version of his message. If he notices that his message would collide with some other message, he may for example wait a number of rounds chosen at random from a suitable distribution before trying to transmit again.

1. Generalizing the Approach

During dinner, the cryptographers also consider how any number of participants greater than one can carry out a version of the protocol. (With two participants, only nonparticipant listeners are unable to distinguish between the two potential senders.) Each participant has a secret key bit in common with, say, every other participant. Each participant outputs the sum, modulo two, of all the key bits he shares, and if he wishes to transmit, he inverts his output. If no participant transmits, the modulo two sum of the outputs must be zero, since every key bit enters exactly twice; if one participant transmits, the sum must be one. (In fact, any even number of transmitting participants yields zero, and any odd number yields one.) For j rounds, each participant could have a j-bit key in common with every other participant, and the ith bit of each such key would be used only in the ith round. Detected collision of messages leads to attempted retransmission as described above; undetected collision results only from an odd number of synchronized identical message segments. (Generalization to fields other than GF(2) is possible, but seems to offer little practical advantage.)

Other generalizations are also considered during dinner. The underlying assumptions are first made explicit, including modeling key-sharing arrangements as graphs. Next, the model is illustrated with some simple examples. The potential for cooperations of participants to violate the security of others is then looked at. Finally, a proof of security based on systems of linear equations is given.

1.1. *Model*

Each participant is assumed to have two kinds of secret: (a) the *keys* shared with other participants for each round; and (b) the *inversion* used in each round (i.e., a 1 if the participant inverts in that round and a 0 if not). Some or all of a participant's secrets may be given to other participants in various forms of collusion, discussion of which is postponed until Section 1.3. (For simplicity in exposition, the possibility of secrets being stolen is ignored throughout.)

The remaining information about the system may be described as: (a) who shares keys with whom; and (b) what each participant *outputs* during each round (the

modulo two sum of that participant's keys and inversion). This information need not be secret to ensure untraceability. If it is publicly known and agreed, it allows various extensions discussed in Sections 2.5 and 2.6. The sum of all the outputs will, of course, usually become known to all participants.

In the terminology of graphs, each participant corresponds to a vertex and each key corresponds to an edge. An edge is incident on the vertices corresponding to the pair of participants that shares the corresponding key. From here on, the graph and dinner-table terminologies will be used interchangeably. Also, without loss of generality, it will be assumed that the graph is connected (i.e., that a path exists between every pair of vertices), since each connected component (i.e., each maximal connected subgraph) could be considered a separate untraceable-sender system.

An *anonymity set seen by a set of keys* is the set of vertices in a connected component of the graph formed from the original graph by removing the edges concerned. Thus a set of keys sees one anonymity set for each connected partition induced by removing the keys. The main theorem of Section 1.4 is essentially that those having only the public information and a set of keys seeing some anonymity set can learn nothing about the members of that anonymity set except the overall parity of their inversions. Thus, for example, any two participants connected by at least one chain of keys unknown to an observer are both in the same anonymity set seen by the observer's keys, and the observer gains nothing that would help distinguish between their messages.

1.2. *Some Examples*

A few simple consequences of the above model may be illustrative. The anonymity set seen by the empty set (i.e., by a nonparticipant observer) is the set of all vertices, since the graph is assumed connected and remains so after zero edges are removed. Also, the anonymity sets seen by the full set of edges are all singleton sets, since each vertex's inversion is just the sum of its output and the corresponding key bits.

If all other participants cooperate fully against one, of course no protocol can keep that singleton's messages untraceable, since untraceability exists only among a set of possible actors, and if the set has only one member, its messages are traceable. For similar reasons, if a participant believes that some subset of other participants will fully cooperate against him, there is no need for him to have keys in common with them.

A biconnected graph (i.e., a graph with at least two vertex-disjoint paths between every pair of vertices) has no cut-vertices (i.e., a single vertex whose removal partitions the graph into disjoint subgraphs). In such a graph, the set of edges incident on a vertex v sees (apart from v) one anonymity set containing all other vertices, since there is a path not containing v between every pair of vertices, and thus they form a connected subgraph excluding v; each participant acting alone learns nothing about the contribution of other participants.

1.3. *Collusion of Participants*

Some participants may cooperate by pooling their keys in efforts to trace the messages of others; such cooperation will be called *collusion*. For simplicity, the

possibilities for multiple collusions or for pooling of information other than full edges will be ignored. Colluders who lie to each other are only touched on briefly, in Section 2.6.

Consider collusion in a complete graph. A vertex is only seen as a singleton anonymity set by the collection of all edges incident on it; all other participants must supply the key they share with a participant in order to determine that participant's inversions. But since a collusion of all but one participant can always trace that participant merely by pooling its members' inversions as already mentioned, it gains nothing more by pooling its keys. The nonsingleton anonymity set seen by all edges incident on a colluding set of vertices in a complete graph is the set of all other vertices; again, a collusion yields nothing more from pooling all its keys than from pooling all its inversions.

Now consider noncomplete graphs. A *full* collusion is a subset of participants pooling all of their keys. The pooled keys see each colluder as a singleton anonymity set; the colluders completely sacrifice the untraceability of their own messages. If a full collusion includes a cut-set of vertices (i.e., one whose removal partitions the graph), the collusion becomes nontrivial because it can learn something about the origin of messages originating outside the collusion; the noncolluding vertices are partitioned into disjoint subgraphs, which are the anonymity sets seen by the pooled keys.

Members of a *partial* collusion pool some but not all of their keys. Unlike the members of a full collusion, each member of a partial collusion in general has a different set of keys. For it to be nontrivial, a partial collusion's pooled keys must include the bridges or separating edges of a segregation or splitting of the graph (i.e., those edges whose removal would partition the graph). Settings are easily constructed in which the pooled keys see anonymity sets that partition the graph and yet leave each colluder in a nonsingleton partition seen by any other participant. Thus, colluders can join a collusion without having to make themselves completely traceable to the collusion's other members.

1.4. *Proof of Security*

Consider, without loss of generality, a single round in which say some full collusion knows some set of keys. Remove the edges known to the collusion from the key-sharing graph and consider any particular connected component C of the remaining graph. The vertices of C thus form an anonymity set seen by the pooled keys.

Informally, what remains to be shown is that the only thing the collusion learns about the members of C is the parity sum of their inversions. This is intuitively apparent, since the inversions of the members of C are each in effect hidden from the collusion by one or more unknown key bits, and only the parity of the sum of these key bits is known (to be zero). Thus the inversions are hidden by a one-time pad, and only their parity is revealed, because only the parity of the pad is known.

The setting is formalized as follows: the connected component C is comprised of m vertices and n edges. The incidence matrix $\mathbf{M}$ of C is defined as usual, with the vertices labeling the rows and the edges labeling the columns. Let K, I, and A be stochastic variables defined on $\mathrm{GF}(2)^n$, $\mathrm{GF}(2)^m$, and $\mathrm{GF}(2)^m$, respectively, such that

K is uniformly distributed over $GF(2)^n$, K and I are mutually independent, and $A = (\mathbf{M}K) \oplus I$. In terms of the protocol, K comprises the keys corresponding to the edges, I consists of the inversions corresponding to the vertices, and A is formed by the outputs of the vertices. Notice that the parity of A (i.e., the modulo two sum of its components) is always equal to the parity of I, since the columns of $\mathbf{M}$ each have zero parity. The desired result is essentially that A reveals no more information about I than the parity of I. More formally:

Theorem. *Let* a *be in* $GF(2)^n$. *For each* i *in* $GF(2)^n$, *which is assumed by* I *with nonzero probability and which has the same parity as* a, *the conditional probability that* $\mathrm{A} = \mathrm{a}$ *given that* $\mathrm{I} = \mathrm{i}$ *is* 2^{1-m}. *Hence, the conditional probability that* $\mathrm{I} = \mathrm{i}$ *given that* $\mathrm{A} = \mathrm{a}$ *is the* a priori *probability that* $\mathrm{I} = \mathrm{i}$.

Proof. Let $i \in GF(2)^n$ have the same parity as a. Consider the system of linear equations $(\mathbf{M}K) \oplus i = a$, in $k \in GF(2)^n$. Since the columns of $\mathbf{M}$ each have even parity, as mentioned above, its rows are linearly dependent over $GF(2)^m$. But as a consequence of the connectedness of the graph, every proper subset of rows of $\mathbf{M}$ is linearly independent. Thus, the rank of $\mathbf{M}$ is $m - 1$, and so each vector with zero parity can be written as a linear combination of the columns of $\mathbf{M}$. This implies that the system is solvable because $i \oplus a$ has even parity. Since the set of n column vectors of $\mathbf{M}$ has rank $m - 1$, the system has exactly 2^{n-m+1} solutions.

Together with the fact that K and I are mutually independent and that K is uniformly distributed, the theorem follows easily. □

2. Some Practical Considerations

After dinner, while discussing how they can continue to make untraceable statements from this respective homes, the cryptographers take up a variety of other topics. In particular, they consider different ways to establish the needed keys; debate adapting the approach to various kinds of communication networks; examine the traditional problems of secrecy and authentication in the context of a system that can provide essentially optimal untraceability; address denial of service caused by malicious and devious participants; and propose means to discourage socially undesirable messages from being sent.

2.1. *Establishing Keys*

One way to provide the keys needed for longer messages is for one member of each pair to toss many coins in advance. Two identical copies of the resulting bits are made, say each on a separate optical disk. Supplying one such disk (which today can hold on the order of 10^{10} bits) to a partner provides enough key bits to allow people to type messages at full speed for years. If participants are not transmitting all the time, the keys can be made to last even longer by using a substantially slower rate when no message is being sent; the full rate would be invoked automatically only when a 1 bit indicated the beginning of a message. (This can also reduce the bandwidth requirements discussed in Section 2.2.)

Another possibility is for a pair to establish a short key and use a cryptographic pseudorandom-sequence generator to expand it as needed. Of course this system might be broken if the generator were broken. Cryptanalysis may be made more difficult, however, by lack of access to the output of individual generators. Even when the cryptographers do not exchange keys at dinner, they can safely do so later using a public-key distribution system (first proposed by [4] and [3]).

2.2 *Underlying Communication Techniques*

A variety of underlying communication networks can be used, and their topology need not be related to that of the key-sharing graph.

Communication systems based on simple cycles, called rings, are common in local area networks. In a typical ring, each node receives each bit and passes it round-robin to the next node. This technology is readily adapted to the present protocols. Consider a single-bit message like the "I paid" message originally sent at the dinner table. Each participant exclusive-or's the bit he receives with his own output before forwarding it to the next participant. When the bit has traveled full circle, it is the exclusive-or sum of all the participants' outputs, which is the desired result of the protocol. To provide these messages to all participants, each bit is sent around a second time by the participant at the end of the loop.

Such an adapted ring requires, on average, a fourfold increase in bandwidth over the obvious traceable protocols in which messages travel only halfway around on average before being taken off the ring by their recipients. Rings differ from the dinner table in that several bit-transmission delays may be required before all the outputs of a particular round are known to all participants; collisions are detected only after such delays.

Efficient use of many other practical communication techniques requires participants to group output bits into blocks. For example, in high-capacity broadcast systems, such as those based on coaxial cable, surface radio, or satellites, more efficient use of channel capacity is obtained by grouping a participant's contribution into a block about the size of a single message (see, e.g., [5]). Use of such communication techniques could require an increase in bandwidth on the order of the number of participants.

In a network with one message per block, the well-known contention protocols can be used: time is divided evenly into frames; a participant transmits a block during one frame; if the block was garbled by collision (presumably with another transmitted block), the participant waits a number of frames chosen at random from some distribution before attempting to retransmit; the participants' waiting intervals may be adjusted on the basis of the collision rate and possibly of other heuristics [5].

In a network with many messages per block, a first block may be used by various anonymous senders to request a "slot reservation" in a second block. A simple scheme would be for each anonymous sender to invert one randomly selected bit in the first block for each slot they wish to reserve in the second block. After the result of the first block becomes known, the participant who caused the ith 1 bit in the first block sends in the ith slot of the second block.

2.3. *Example Key-Sharing Graphs*

In large systems it may be desirable to use fewer than the $m(m-1)/2$ keys required by a complete graph. If the graph is merely a cycle, then individuals acting alone learn nothing, but any two colluders can partition the graph, perhaps fully compromising a participant immediately between them. Such a topology might nevertheless be adequate in an application in which nearby participants are not likely to collude against one another.

A different topology assumes the existence of a subset of participants who each participant believes are sufficiently unlikely to collude, such as participants with conflicting interests. This subset constitutes a fully connected subgraph, and the other participants each share a key with every member of it. Every participant is then untraceable among all the others, unless all members of the completely connected subset cooperate. (Such a situation is mentioned again in Section 3.)

If many people wish to participate in an untraceable communication system, hierarchical arrangements may offer further economy of keys. Consider an example in which a representative from each local fully connected subgraph is also a member of the fully connected central subgraph. The nonrepresentative members of a local subgraph provide the sum of their outputs to their representative. Representatives would then add their own contributions before providing the sum to the central subgraph. Only a local subgraph's representative, or a collusion of representatives from all other local subgraphs, can recognize messages as coming from the local subgraph. A collusion comprising the representative and all but one nonrepresentative member of a local subgraph is needed for messages to be recognized as coming from the remaining member.

2.4. *Secrecy and Authentication*

What about the usual cryptologic problems of secrecy and authentication?

A cryptographer can ensure the secrecy of an anonymous message by encrypting the message with the intended recipient's public key. (The message should include a hundred or so random bits to foil attempts to confirm a guess at its content [1].) The sender can even keep the identity of the intended recipient secret by leaving it to each recipient to try to decrypt every message. Alternatively, a prearranged prefix could be attached to each message so that the recipient need only decrypt messages with recognized prefixes. To keep even the multiplicity of a prefix's use from being revealed, a different prefix might be used each time. New prefixes could be agreed in advance, generated cryptographically as needed, or supplied in earlier messages.

Authentication is also quite useful in systems without identification. Even though the messages are untraceable, they might still bear digital signatures corresponding to public-key "digital pseudonyms" [1]; only the untraceable owner of such a pseudonym would be able to sign subsequent messages with it. Secure payment protocols have elsewhere been proposed in which the payer and/or the payee might be untraceable [2]. Other protocols have been proposed that allow individuals known only by pseudonyms to transfer securely information about themselves between organizations [2]. All these systems require solutions to the sender untrace-

ability problem, such as the solution presented here, if they are to protect the unlinkability of pseudonyms used to conduct transactions from home.

2.5. *Disruption*

Another question is how to stop participants who, accidentally or even intentionally, *disrupt* the system by preventing others from sending messages. In a sense, this problem has no solution, since any participant can send messages continuously, thereby clogging the channel. But nondisupters can ultimately stop disruption in a system meeting the following requirements: (1) the key-sharing graph is publicly agreed on; (2) each participant's outputs are publicly agreed on in such a way that participants cannot change their output for a round on the basis of other participants' outputs for that round; and (3) some rounds contain inversions that would not compromise the untraceability of any nondisrupter.

The first requirement has already been mentioned in Section 1.1, where it was said that this information need not be secret; now it is required that this information actually be made known to all participants and that the participants agree on it.

The second requirement is in part that disrupters be unable (at least with some significant probability) to change their output after hearing other participants' outputs. Some actual channels would automatically ensure this, such as broadcast systems in which all broadcasts are made simultaneously on different frequencies. The remainder of the second requirement, that the outputs be publicly agreed on, might also be met by broadcasting. Having only channels that do not provide it automatically, an effective way to meet the full second requirement would be for participants to "commit" to their outputs before making them. One way to do this is for participants to make public and agree on some (possibly compressing and hierarchical, see Section 2.6) one-way function of their outputs, before the outputs are made public.

The third requirement is that at least some rounds can be *contested* (i.e., that all inversions can be made public) without compromising the untraceability of nondisrupting senders. The feasibility of this will be demonstrated here by a simple example protocol based on the slot reservation technique already described in Section 2.2.

Suppose that each participant is always to make a single reservation in each reserving block, whether or not he actually intends to send a message. (Notice that, because of the "birthday paradox," the number of bits per reserving block must be quadratic in the number of participants.) A disrupted reserving block would then with very high probability have Hamming weight unequal to the number of participants. All bits of such a disrupted reserving block could be contested without loss of untraceability for nondisrupters.

The reserved blocks can also be made to have such safely contestable bits if participants send trap messages. To lay a trap, a participant first chooses the index of a bit in some reserving block, a random message, and a secret key. Then the trapper makes public an encryption, using the secret key, of both the bit index and the random message. Later, the trapper reserves by inverting in the round corresponding to the bit index, and sends the random message in the resulting reserved

slot. If a disrupter is unlucky enough to have damaged a trap message, then release of the secret key by the trapper would cause at least one bit of the reserved slot to be contested.

With the three requirements satisfied, it remains to be shown how if enough disrupted rounds are contested, the disrupters will be excluded from the network.

Consider first the case of a single participant's mail computer disrupting the network. If it tells the truth about contested key bits it shares (or lies about an even number of bits), the disrupter implicates itself, because its contribution to the sum is unequal to the sum of these bits (apart from any allowed inversion). If, on the other hand, the single disrupter lies about some odd number of shared bits, the values it claims will differ from those claimed for the same shared bits by the other participants sharing them. The disrupter thereby casts suspicion on all participants, including itself, that share the disputed bits. (It may be difficult for a disrupter to cast substantial suspicion on a large set of participants, since all the disputed bits will be in common with the disrupter.) Notice, however, that participants who have been falsely accused will know that they have been—and by whom—and should at least refuse to share bits with the disrupter in the future.

Even with colluding multiple disrupters, at least one inversion must be revealed as illegitimate or at least one key bit disputed, since the parity of the outputs does not correspond to the number of legitimate inversions. The result of such a contested round will be the removal of at least one edge or at least one vertex from the agreed graph. Thus, if every disruptive action has a nonzero probability of being contested, only a bounded amount of disruption is possible before the disrupters share no keys with anyone in the network, or before they are revealed, and are in either case excluded from the network.

The extension presented next can demonstrate the true value of disputed bits, and hence allows direct incrimination of disrupters.

2.6. *Tracing by Consent*

Antisocial use of a network can be deterred if the cooperation of most participants makes it possible, albeit expensive, to trace any message. If, for example, a threatening message is sent, a court might order all participants to reveal their shared key bits for a round of the message. The sender of the offending message might try to spread the blame, however, by lying about some odd number of shared bits. Digital signatures can be used to stop such blame-spreading altogether. In principle, each party sharing a key could insist on a signature, made by the other party sharing, for the value of each shared bit.

Such signatures would allow for contested rounds to be fully resolved, for accused senders to exonerate themselves, and even for colluders to convince each other that they are pooling true keys. Unfortunately, cooperating participants able to trace a message to its sender could convince others of the message's origin by revealing the sender's own signatures. A variation can prevent a participant's signatures from being used against him in this way: instead of each member of a pair of participants signing the same shared key bit, each signs a separate bit, such that the sum of the signed bits is the actual shared key bit. Signatures on such "split"

key bits would still be useful in resolving contested rounds, since if one contester of a bit shows a signature made by the second contester, then the second would have to reveal the corresponding signature made by the first or be thought to be a disrupter.

In many applications it may be impractical to obtain a separate signature on every key bit or split key bit. The overhead involved could be greatly reduced, however, by digitally signing cryptographic compressions of large numbers of key bits. This might of course require that a whole block of key bits be exposed in showing a signature, but such blocks could be padded with cryptographically generated pseudorandom (or truly random) bits, to allow the exposure of fewer bits per signature. The number of bits and amount of time required to verify a signature for a single bit can be reduced further by using a rooted tree in which each node is the one-way compression function of all its direct descendants; only a digital signature of each participant's root need be agreed on before use of the keys comprising the leaves.

3. Relation to Previous Work

There is another multiparty-secure sender-untraceability protocol in the literature [1]. To facilitate comparison, it will be called a *mix-net* here, while the protocol of the present work is called a *dc-net*. The mix-net approach relies on the security of a true public-key system (and possibly also of a conventional cryptosystem), and is thus at best computationally secure; the dc-net approach can use unconditional secrecy channels to provide an unconditionally secure untraceable-sender system, or can use public-key distribution to provide a computationally secure system (as described in Section 2.1).

Under some trust assumptions and channel limitations, however, mix-nets can operate where dc-nets cannot. Suppose that a subset of participants is trusted by every other participant not to collude and that the bandwidth of at least some participants' channels to the trusted subset is incapable of handling the total message traffic. Then mix-nets may operate quite satisfactorily, but dc-nets will be unable to protect fully each participant's untraceability. Mix-nets can also provide recipient untraceability in this communication environment, even though there is insufficient bandwidth for use of the broadcast approach (mentioned in Section 2.4).

If optimal protection against collusion is to be provided and the crypto-security of mix-nets is acceptable, a choice between mix-nets and dc-nets may depend on the nature of the traffic. With a mail-like system that requires only periodic deliveries, and where the average number of messages per interval is relatively large, mix-nets may be suitable. When messages must be delivered continually and there is no time for batching large numbers of them, dc-nets appear preferable.

4. Conclusion

This solution to the dining cryptographers problem demonstrates that unconditional secrecy channels can be used to construct an unconditional sender-untraceability channel. It also shows that a public-key distribution system can be used to

construct a computationally secure sender-untraceability channel. The approach appears able to satisfy a wide range of practical concerns.

Acknowledgments

I am pleased to thank Jurjen Bos, Gilles Brassard, Jan-Hendrik Evertse, and the untraceable referees for all their help in revising this article. It is also a pleasure to thank, as in the original version that was distributed at Crypto 84, Whitfield Diffie, Ron Rivest, and Gus Simmons for some stimulating dinner-table conversations.

References

[1] Chaum, D., Untraceable Electronic Mail, Return Addresses, and Digital Pseudonyms, *Communications of the ACM*, vol. 24, no. 2, February 1981, pp. 84–88.

[2] Chaum, D., Security Without Identification: Transaction Systems to Make Big Brother Obsolete, *Communications of the ACM*, vol. 28, no. 10, October 1985, pp. 1030–1044.

[3] Diffie, W., and Hellman, M. E., New Directions in Cryptography, *IEEE Transactions on Information Theory*, vol. 22, no. 6, November 1976, pp. 644–654.

[4] Merkle, R. C., Secure Communication over Insecure Channels, *Communications of the ACM*, vol. 21, no. 4, 1978, pp. 294–299.

[5] Tanenbaum, A. S., *Computer Networks*, Prentice Hall, Englewood Cliffs, New Jersey, 1981.