UNIVERSITÄT KARLSRUHE
FAKULTÄT FÜR INFORMATIK
Postfach 6880, D 7500 Karlsruhe 1

How to implement ISDNs without user observability — Some remarks

Andreas Pfitzmann

Institut für Informatik IV, Universität Karlsruhe, Postfach 6980,
D 7500 Karlsruhe 1, West Germany

Interner Bericht 14/85

Abstract

In present day communication networks, the network operator or an intruder could easily observe when, how much and with whom the users communicate (traffic analysis), even if the users employ end-to-end encryption. With the increasing use of ISDNs, this becomes a severe threat.
Therefore, we summarize basic concepts to keep the recipient and sender or at least their relationship unobservable, consider some possible implementations and necessary hierarchical extensions, and propose and evaluate some suitable performance and reliability enhancements.

Keywords

traffic analysis, user observability, anonymity, fault tolerance, ISDN, MIX-network, DC-network, RING-network, switched/broadcast network, broadcast/broadcast network, tree network with dynamic key graphs, switched/tree network, Trojan Horse, confinement problem, covert channel

CR Categories

C.2 COMPUTER-COMMUNICATION NETWORKS;
GENERAL; Security and protection
NETWORK ARCHITECTURE AND DESIGN; Network topology
NETWORK PROTOCOLS; Protocol architecture
LOCAL NETWORKS; Access schemes, Rings
D.4.6 SECURITY AND PROTECTION; Information flow controls
E.3 DATA ENCRYPTION; Public key cryptosystems
H.4.3 INFORMATION SYSTEMS APPLICATIONS;
COMMUNICATIONS APPLICATIONS; Electronic mail, Videotex
K.4.1 COMPUTING MILLIEUX; COMPUTERS AND SOCIETY;
PUBLIC POLICY ISSUES; Privacy

Table of contents

Table of figures

0 Motivation

Public and private communication networks have a growing importance for our daily life. We use them for telephony, telegraphy, television, videotex, radio and in the near future we will use them for video telephony, electronic mail, ordering and receiving of newspapers, home banking, etc.
All these services will be integrated in a so-called Integrated Services Digital Network (ISDN). If such a network is built as planned e.g. by the german PTT and operated on a "transmission on demand" basis even for the classical broadcast services TV and radio, major parts of any user's life might easily be observed by the PTT or by an intruder [Pfit_83, Pfit_84, Pfit_85, PfPW_86].
Eavesdropping can be foiled by link-by-link encryption [Bara_64], but this does not foil attackers at the stations (e.g. via Trojan Horses).
There are some well known measures allowing users themselves to decrease their observability. The content of a message can be sufficiently hidden by end-to-end encryption. However, an attacker can still observe who sends how many messages to whom and at what time (traffic analysis). To hide this information, too, they can use public network stations (e.g. telephone boxes) instead of private ones. This will prevent observation but is very uncomfortable for the users (e.g. who would watch TV in a video telephone box?). If they use their own private network stations, they can only try to hide their behaviour by making their network stations do more things than necessary or do necessary things before they are actually needed. For example, a user can order a whole newspaper or several newspapers instead of a single article, and he can do so at any time before he wants to read them. This is an easy but expensive measure and not suitable for services like telephony.

So the only way to decrease user observability in a comfortable and cheap fashion seems to be to design a network for anonymity and not to try to achieve anonymity afterwards.
Of course, the standard requirements for any network, i.e.

performance and reliability, have to be met, too. Since these requirements are particularly strong for ISDNs, we will mostly discuss them in the ISDN context. Where it may be appropriate to trade unobservability versus performance or reliability for a special purpose network or for some part of the bandwidth of an ISDN, we will note that (cf. sections 2, 2.3, 2.4).

In the following we will describe

* the existing basic concepts for anonymous networks (i.e. concepts enabling unobservable reception, unlinkability of sender and recipient or anonymous sending) in a systematic way in section 1,
* suitable implementations and some extensions to meet stringent requirements on performance (i.e. anonymous channel switching, suitable implementations of the extended basic concepts, heterogeneous networks and hierarchical ones) in section 2,
* some extensions to tolerate faults in order to meet stringent requirements on reliability in section 3,
* who should establish which parts of the network and be responsible for the quality of service in section 4,
* some conclusions in section 5, and
* an application of the techniques to the confinement problem.

1 Basic concepts for anonymous networks

1.1 A closer look at anonymity

What we would like to realize is absolute anonymity against every possible attacker. But an attacker could control all network stations, all lines, and even the communication partner (e.g. an insurance company) and so absolute anonymity is impossible. Therefore, we first need reasonable models of possible attackers.

There are several possible attackers: the administration, foreign states, companies, one's neighbors and communication partners. During the design of an anonymous network these possible attackers have to be translated into terms of stations and lines. A station is always under control of its owner (we try to avoid stations which reside, at least partially, in so called "tamper-resistant modules" [Kent_80, Chau_84, DaPr_84 p. 3, 80, Simm_85, HeFi_80], because their security will never be beyond doubt, they are inconvenient to maintain, and they are costly) and might be under control of everybody who has had access to it so far, e.g. its manufacturer, because he might have installed a Trojan Horse [PoKl_78, Thom_84]. Trojan Horses are a serious problem in stations with high complexity, e.g. switching centers. In simple user stations they can be detected (if tried) more easily. Lines are assumed to be owned by the PTT. Normally they can easily be observed by the PTT (since it operates equipment connected to these lines) or by an eavesdropper, but by physical measures (no equipment connected to these lines is operated by the PTT, and the cable is run on public ground as seldom as possible) such an attack can be made much more difficult.

Given a model of the attacker we secondly have to define what events we want to keep hidden from him. A strong possibility is to keep the sending and receiving of a message secret (sender and recipient anonymity). A weaker possibility is to keep only the relationship between sender and recipient secret, i.e.

sending and receiving of physical messages is observable, but it is infeasable for an attacker to link the physical message sent by the sender and the physical message received by the recipient. This concept of unlinkability is also weaker than sender or recipient anonymity alone, because an attacker who cannot determine e.g. the recipient of a message also cannot determine the relation between the sender of the message and this recipient.

A third criterion for anonymity is among how many alternatives something is hidden from the attacker, a fourth whether it is hidden in an absolute, information-theoretic sense (i.e. the attacker can compute as long as he likes but never finds out what should be hidden from him) or only in the sense of computational complexity (and thus not provably nowadays).

As most concepts for anonymous networks hide the same kind of events from all attackers considered reasonable for that concept (and there is complete anonymity against some weaker attackers and no anonymity against stronger attackers) we use this criterion to arrange the concepts.

1.2 Recipient anonymity

Receiving a message can be made completely anonymous to the network by delivering the message to all stations (broadcast). If the message has an intended recipient, a so called addressee, it has to contain an attribute by which he and nobody else can recognize it as addressed to him [FaLa_75]. This attribute is called an implicit address in contrast to an explicit address, which describes a place in the network.

Implicit addresses can be distinguished according to their visibility, i.e. whether they can be tested for equality or not. An implicit address is called invisible, if it is only visible to its addressee and is called visible otherwise [Waid_85].
Invisible implicit addresses can be realized with a public key

cryptosystem. A message is addressed by encrypting it (or a part of it) with a public key of the addressee's station. Each station decrypts all messages with each of its private keys and uses the message redundancy to decide which messages are addressed to it.
Conversely, any invisible addressing scheme for messages can be used for public key distribution: If A wants to communicate an n bit key to B, A chooses n messages randomly, and addresses them to B if the corresponding key bit is 1, and addresses them not to B otherwise. A sends these n messages in one explicitly addressed message to B. (Please note that messages are considered to be independent entities. Therefore attributes associated with them, e.g. time of transmission cannot be used for the purpose of addressing. This is in contrast to channel switching, which will be discussed in section 2.1.)
This addressee-anonymity-based public key distribution protocol corresponds to a sender-anonymity-based one proposed by Alpern and Schneider [AlSc_83].
If a secret key of a faster conventional cryptosystem has already been exchanged, that key can also be used for invisible implicit addressing [Karg_77 pp. 111..112]. If this is done to save decipherment cost, each packet should start with a bit telling which cryptosystem is used for invisible implicit addressing.
Visible implicit addresses can be realized much easier: Users choose arbitrary names for themselves, which can then be prefixed to messages.

Another criterion to distinguish implicit addresses is their distribution (see figure 1). An implicit address is called public, if it is known to every user (like telephone numbers today) and private if the sender received it secretly from the addressee either outside the network or as a return address or by a generating algorithm the sender and the addressee agreed upon [FaLa_75, Karg_77].
Public addresses should not be realized by visible implicit addresses to avoid the linkability of the visible public address of a message and the addressed user.

Private addresses can be realized by visible addresses but then each of them should be used only once.

Figure 1 gives an assessment of the combinations of addressing modes and address distribution concerning the unobservability of the recipient and the expense necessary. Note that, whereas the anonymity of a recipient of a message provided by broadcast without addresses or with private ones is possible in the information-theoretic sense, the unlinkability of a message addressed with a public invisible address and the holder of this address (more precisely: that pseudonym of the holder under which the address is published) is only possible in the sense of computational complexity.

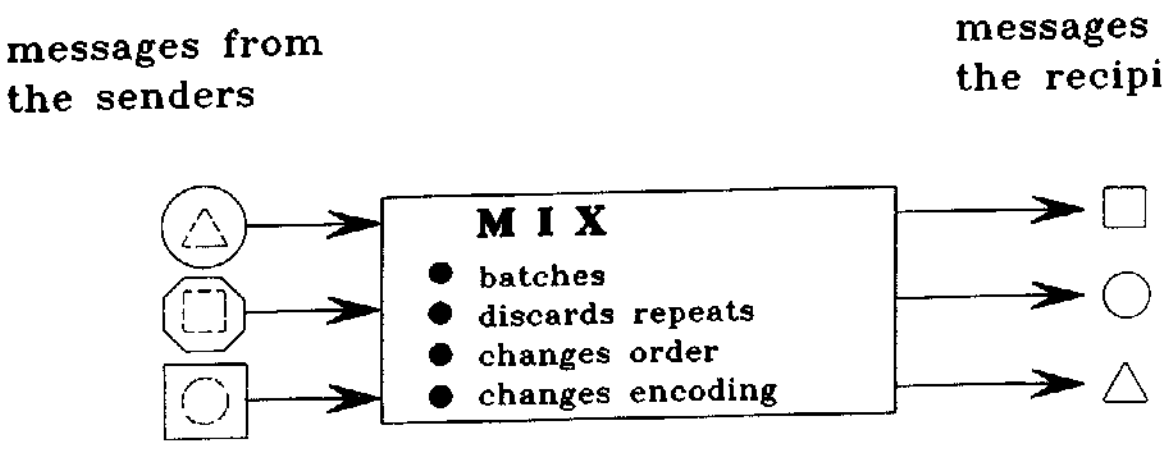
			address distribution	
			public address	private address
addressing modes	implicit address	invisible	very costly, but necessary to establish contact	costly
		visible	linkable to (anonymous) holder	change after use
	explicit address		linkable to holder	linkable to holder

Fig. 1: Combinations of addressing modes and address distribution

1.3 Unlinkability of sender and recipient (MIX-network)

This form of anonymity can be realized by a special network station, a so called MIX, which collects a batch, i.e. a number of messages (of equal length) from many distinct senders,

discards repeats, changes the encodings of messages and forwards them to the recipients in a different order [Chau_81]. Figure 2 gives a pictorial and a formal representation.
The MIX must collect messages from many distinct senders, since if an attacker submits n messages to the MIX and the MIX uses only n+1 messages in its batch, the attacker can bridge the MIX with respect to all n+1 messages.
If the MIX did not discard repeated messages (as long as it does not change its key pair, which determines the change of encoding) an attacker could draw conclusions, since a repeat of an input message would cause a repeat of the corresponding output message.



Change of encoding of a message can be implemented using a public-key cryptosystem (e_{MIX} public-key for encipherment of the MIX, d_{MIX} corresponding private key for decipherment) and random bit strings r_i

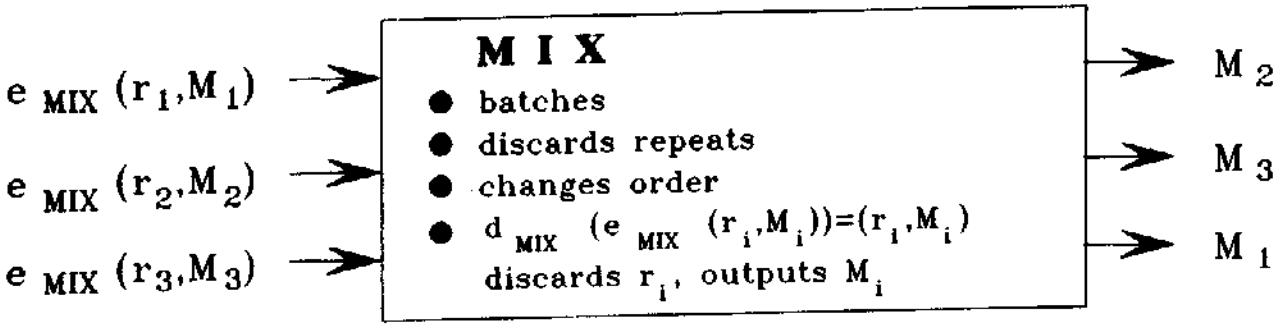


Fig. 2: MIX hides the relation between its in- and output messages

This measure hides the relation between sender and recipient of a message from everybody but the MIX and the sender of the message (in the sense of computational complexity and among as many possibilities as there were different senders and recipients of messages of this batch).

By using more than one MIX to forward a message from the sender to the recipient, the relation is hidden from every attacker in the network who does not control all MIXes which the message passed, nor has the cooperation of the sender of the message under consideration.
The complete scheme how the sender must encode the message so that it can pass n MIXes is as follows: Let $A_1,\ldots,A_n$ be the sequence of addresses and $e_1,\ldots,e_n$ the sequence of public keys of the chosen MIXes $MIX_1,\ldots,MIX_n$; $r_1,\ldots,r_n$ the chosen sequence of random bit strings; A_{n+1} the address of the addressee (called MIX_{n+1} for convenience), and e_{n+1} his public key. The sender forms the messages M_i, which MIX_i shall receive, according to the following recursive scheme, starting from the message content M that MIX_{n+1} shall receive:

$$M_{n+1} = e_{n+1}(M) \qquad \text{(pure scheme)}$$
$$M_i = e_i(r_i, A_{i+1}, M_{i+1}) \quad \text{for } i=1,\ldots,n$$

Then he sends M_1 to MIX_1.

By combining this scheme and that of section 1.2, even the sender cannot learn who receives his message.
Another, but weaker possibility is that every MIX attaches a random bit string to each message and encrypts it with e.g. a public key of the next MIX or of the addressee, respectively (virtual-link-by-virtual-link encryption). This prevents even the sender of a message, who is supposed to tap all lines, from tracing his message on its way to the addressee, if he has not the cooperation of the last MIX.

To enable the recipient to reply (while keeping the original sender anonymous and without requiring broadcast), there is a

scheme for untraceable return addresses, which the sender can form and deliver to the recipient in his message.
Such a return address looks like a message to the sender in the original scheme, except for two differences.
The first is that of course there is no message content (or perhaps just some number by which the sender can distinguish different responses when they come back).
The second difference stems from the fact that the recipient cannot put the message content into the return address, so he must send the return address and the message content side by side. To enable the MIXes to change not only the outlook of the address part but that of the message part as well, the random bit strings in the return address are chosen as random keys of some cryptosystem (which can be a public or private key system), and each MIX does not discard the random key it finds but uses it to encipher the message part (there is no need to include a random bit string there, because nobody else knows the key, which would be necessary to compare input and output).
One more key is sent to the recipient together with the return address, by which he enciphers the return message content.
So the original sender gets the reply message enciphered with all the keys he chose and therefore he can decipher it, but nobody else. His relation to the recipient (= sender of the reply message) is as well protected as if the original sender had sent a message to the recipient using the same sequence of MIXes.

Formally the scheme looks like this: Let $MIX_1,\ldots,MIX_m$ be the chosen sequence of MIXes (in the order in which they will mix the reply message) and call the original sender = addressee of the reply message MIX_{m+1} for convenience. Let e_j be the public key and A_j the address of MIX_j and k_j the key chosen for this occasion for MIX_j.
The original sender sends the <u>return address</u> (k_0, A_1, B_1) to the recipient, where k_0 is the key chosen for encryption of the reply message and B_1 the actual return address part, which is formed according to the following recursive scheme, where each B_j is the return address part that MIX_j will receive with the

reply message:

$B_{m+1} = \emptyset$ (return address scheme a)
$B_j = e_j(k_j, A_{j+1}, B_{j+1})$ for $j=1,\ldots,m$

The reply messages M_j that MIX_j shall receive are derived from the reply message content C by the recursion

$M_1 = B_1, C_1; \quad C_1 = k_0(C)$ (return address scheme b)
$M_j = B_j, C_j; \quad C_j = k_{j-1}(C_{j-1})$ for $j=2,\ldots,m+1$

So the original sender receives $M_{m+1} = k_m(\ldots k_1(k_0(C))\ldots)$.

Of course, in the return address scheme, MIXes must not allow address parts to be repeated. The repetition of a message part seems uncritical: If an attacker wants to trace $M_i = B_i, C_i$ over MIX_i by submitting a second message $M_i' = B_i', C_i$ to MIX_i in the hope that MIX_i also outputs two messages with the same message part C_{i+1}, then MIX_i must find the same private key k_i in both B_i and B_i'. But the attacker does not know k_i and should not be able to change a single bit in B_i without B_i decrypting to garbage.

To a certain degree, virtual-link-by-virtual-link encryption can keep the recipient anonymous from the original sender, who needs the cooperation of the MIX most close to the recipient to identify him. But since the original sender may choose this MIX at will, we have to suppose he has this cooperation.
Therefore, a cautious recipient will enclose his reply message (untraceable return address and encrypted reply message content) in an untraceable message according to the pure scheme to some MIX, so that that MIX has to use the untraceable return address first, which gives no information about the recipient even to the original sender.

With this use of return addresses, the scheme is more secure than the original one, because both sender and recipient can be kept anonymous against each other. So it seems reasonable that

already the first message of a connection should also be of this kind, i.e. the addresses found in address directories should not be physical addresses of user stations, which the sender uses to form a message according to the pure scheme, but anonymous return addresses. Of course, then each entry of the address directory must be a huge sequence of anonymous return addresses, since each address must not be used more than once. This is of course clumsy for printed directories but no problem for electronic ones which issue each untraceable return address only once.

We should also mention that in the schemes described so far, the messages get shorter at each MIX, because the random bits are discarded, so that the current lengths of messages give some information to an attacker. But there is also a scheme which ensures that a message stays as long as it is at the first MIX until the recipient gets it, see [Chau_81 p. 87].

1.4 Sender anonymity

A very limited possibility of achieving sender anonymity (nevertheless the one still most often cited in the network oriented literature) is that each user station generates dummy traffic, i.e. it sends more messages than it really has to [VoKe_83 p. 157, 158, VoKe_85 p. 18].
Then an attacker cannot notice when a user station really has something to send and how much. But the attacker may notice that a user station has nothing to send, if the user station is quiet, which will be the case most of the time if the user stations share communication resources.
A greater problem is that our attacker knows the sender, if he is the recipient of a message.
Of course dummy traffic is a symmetric concept in the sense that it provides as much anonymity to the receivers of the dummy messages as to their senders. We only did not mention it in section 1.2 as a recipient anonymity scheme because using messages addressed to other stations instead of dummy messages

is not more expensive and provides anonymity also against the sender of the message, so broadcast (or at least multicast, cf. section 2.4) is a more reasonable choice.

The MIX-network can also be regarded as a sender anonymity scheme if every station acts as MIX, because then the message a station sends is hidden among the messages of a complete batch, though the fact that a message has been sent can still be seen by comparing the number of input and output messages, if no message of the same batch is received by this station and as long as no dummy traffic is added.
Symmetrically in this case the MIX-network also provides recipient anonymity if untraceable return addresses are used, and at least recipient anonymity against every attacker who is not the sender of the message if the pure scheme is used, or against every attacker who is not both the sender of the message and controls the last MIX if the pure scheme and virtual-link-by-virtual-link encryption are used.
Anyway the security of the MIX-network is only in the sense of computational complexity, and the MIX-scheme with all stations as MIXes is at least difficult to realize, if not impossible for some services, cf. section 2.2.3.

1.4.1 Superposed sending (DC-network)

A powerful scheme for sender anonymity is superposed sending which is published in [Cha3_85, Cha8_85] and is called DC-network (dining cryptographers network) there. It is somehow symmetric to broadcast: There all stations get a message (possibly concurrently) to enable one station to receive it anonymously, here all stations output a message (possibly concurrently) so that a single station may send one anonymously. This is done in the following way:
Each user station generates at least one key bit for each message bit and sends each key bit to exactly one other user station over a secure channel. To send one bit every user station adds modulo 2 (superposes) all generated and received

key bits and its message bit if there is one. The sums are sent over the network and added up modulo 2. The result is distributed to all user stations. It is the sum of all sent message bits, because every key bit was added twice (figure 3).

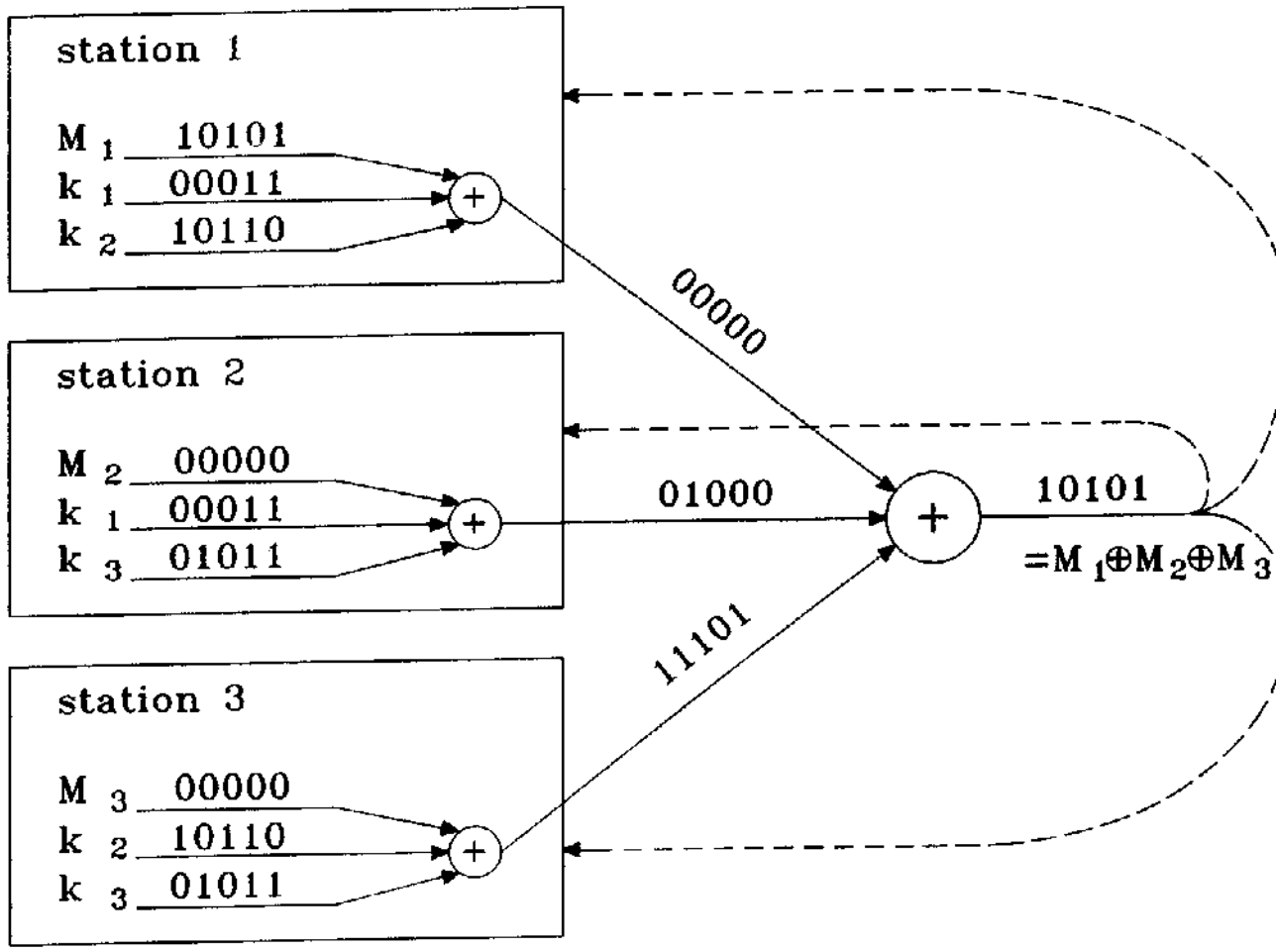


Fig. 3: Three stations superposing exchanged keys k_i and messages M_j

Therefore, the scheme realizes a multi-access channel with collisions. For its efficient use a medium access protocol [Tane_81] preserving anonymity is needed. Two of them are mentioned in [Cha3_85]: slotted ALOHA and an anonymous reservation scheme which reduces the probability of collisions by using the inversion of one bit (out of many bits exclusively used for reservations) to signal a reservation [Tane_81 p. 272].

If an attacker controls all lines and some of the user stations, he gets no information about the sender of a message among the other users, as long as their key graph, i.e. the graph with the users as nodes and the keys as edges, is connected.
Superposed sending requires the exchange of a tremendous amount of randomly chosen keys. To reduce costs, pseudorandomly generated keys can be used instead, reducing information-theoretic security (or more precisely: perfect anonymity corresponding to perfect secrecy [Shan_49 p. 659, Denn_82 p. 22]) to computational security [DiHe_76 p. 646, Denn_82 p. 3, Yao_82]. To maintain security (especially if the attacker is sometimes allowed to see the outcome of single pseudorandom bit generators during the tracing of protocol violators or faults, cf. section 3.2), cryptographically strong pseudorandom bit generators [BlMi_84, VaVa_85] have to be used.

1.4.2 Physical unobservability (RING-network)

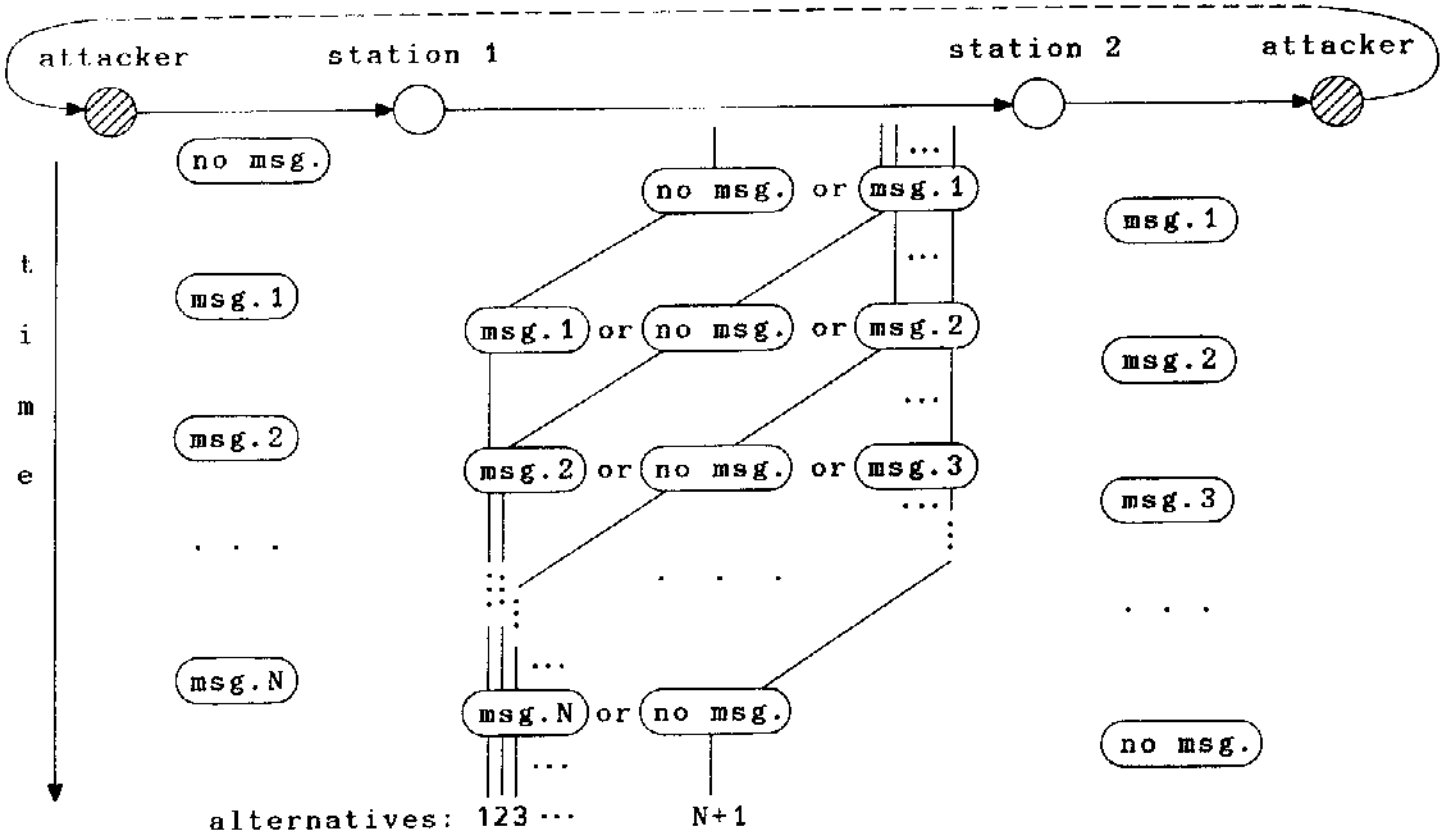
The expensive generation, distribution and superposition of keys (and messages) of the concept of superposed sending can be avoided if the network is designed for preventing attackers from physically observing all lines connecting a user with the rest of the world (in contrast to a conventional star network, where all lines and hence all users can be observed by the switching center). So we will assume a less capable attacker than in the DC-network, but make this assumption reasonable by means of the physical arrangement of the network.
A simple and efficient way to do so is to connect the user stations by RINGs, which are in wide use for local area networks. If an anonymous medium access protocol is used, the sending of a user station is only observable if its two neighbor stations collude or the lines are tapped. The latter attack can be prevented by running the cable in an appropriate way [Pfi1_83, Pfit_84]. Then, it is approximately as difficult to observe the sending of a user station as to observe its owner's behavior at home directly by hidden microphones, laser-based bugs or EMI emissions of the user station [Horg_85, Hor2_85,

Eck_85]. Even if the sending of a station is observed by an attacker, he needs the cooperation of the recipient to observe who is communicating with whom, since the RING-network broadcasts all messages to all stations yielding complete recipient anonymity.

Suitable medium access protocols for the RING-network are slotted ring with sender remove and token ring, both with exhaustive service [Höck_85, HöPf_85]. This is illustrated in figure 4.
These two medium access protocols implement distributed and unobservable polling which prevents collisions. Therefore, a ring utilization near 1 can be reached. The disadvantage of these two medium access protocols is that the right to send can only be passed in one direction, which restricts the number of alternatives, whose station can have sent which message. This is shown in figure 4.
A random access protocol with collisions like slotted ALOHA [Tane_81] avoids this restriction and is therefore better from an unobservability point of view. But the achievable ring utilization drops to $1/e$ (≈ 0.37) and in the case that the content of a slot does not return to its sender (because another station overwrote it), the sender is in doubt whether the addressee got it or not.

Since for low ring utilization the difference in the number of alternatives is small and even disappears for very low ring utilizations, and for high ring utilization slotted ALOHA cannot be used at all, we recommend to use the distributed polling schemes.

In any case the security provided by this scheme is information-theoretic, though in most cases there are fewer alternatives for every event than in the DC-network (this sort of security corresponds to the kind of secrecy called ideal in [Shan_49 p. 660] and unconditional in [Denn_82 pp. 3, 25]).



The most efficient ring access protocols (token ring, slotted ring) pass the right to send (RTS) around the ring.
If an attacker surrounds 2 stations, which send N messages after RTS is passed to them and then pass RTS, there are N+1 alternatives which station can have sent which message (msg.).
For every message, there is at least one alternative, in which station i (i = 1, 2) sent the message.

The example can be generalized to g surrounded stations.
There are $\binom{N+g-1}{N}$ alternatives and there is at least one alternative for every message, in which station i (i = 1, 2, ..., g) sent it.

Fig. 4: An anonymous medium access protocol for RING-networks guarantees that an attacker surrounding a sequence of stations cannot detect which one has been the sender of a particular message

1.5 Layering

To facilitate the design, understanding, implementation and interconnection of networks, they are designed as a sequence of layers. Every layer uses the services of the lower layer to provide to the upper layer a more comfortable service. The International Standards Organization (ISO) standardized a seven layer model, the "Basic reference model for Open System Interconnection" (OSI) [OSI_83, DaZi_83], whose layers are refined when necessary, e.g. to accommodate it to local area networks (LANs) [Finl_84, Kühn_85, Agne_85, Blan_84].

A lot of useful comments on the layering of systems to facilitate their design can be found in [Parn_74, Röhr_82]. As noted in [Parn_74 p. 337], this (design) layering must be distinguished from the (implementation) layering of the "running system", called interpretation system ("Interpretationssaystem") in [Gilo_81 p. 12, 13].

Figure 5 presents a synopsis of the MIX-, DC- and RING-network concerning their (design) layering in the framework of OSI. Special emphasis is placed on the lowest layer boundary where anonymous communication is provided.
As shown, the MIX-network may use arbitrary implementations of medium, physical layer, and data link layer and the DC-network may use an arbitrary medium. These layers of the MIX- and DC-network can be implemented without restrictions caused by anonymity requirements. All measures to enhance performance and reliability can be used.
On top of these layers, the MIX- and DC-network employ special cryptographic mechanisms to create anonymity, which we inserted in the (upper sublayers of the) network and physical layer respectively. The RING-network employs as medium a ring, which together with digital signal (re)generation provides anonymous communication.

OSI layers	MIX-network	DC-network	RING-network
application layer	preserving anonymity	preserving anonymity	preserving anonymity
presentation layer	preserving anonymity	preserving anonymity	preserving anonymity
session layer	preserving anonymity	preserving anonymity	preserving anonymity
transport layer	preserving anonymity	preserving anonymity	preserving anonymity
network layer	*anonymous* MIX	preserving anonymity	preserving anonymity
data link layer	arbitrary	anonymous medium access	anonymous medium access
physical layer	arbitrary	*anonymous* superposed sending	*anonymous* digital signal (re)generation
medium	arbitrary	arbitrary	ring

Fig. 5: Synopsis of the layering of MIX-, DC- and RING-network in the framework of OSI

All layers on top of the layer(s) providing anonymity must preserve it. That means, that no (or at least: not too many) alternatives possible at the (design) layer providing anonymity are excluded by the protocols of the (design) layers above. Otherwise, anonymous communication becomes fiction. For example, the medium access protocols, which should efficiently manage the bandwidth of the multi-access channel of the DC- and RING-network, must preserve anonymity.

Of course, the special mechanisms of the MIX-, DC- and RING-network can be implemented in higher (design) layers as well, e.g. on top of an arbitrarily designed network layer. MIXes could then be embedded in the transport layer, which then had to contain some functions of the network layer a second time, e.g. routing. If superposed sending is embedded in the transport layer, even anonymous medium access must be embedded in the higher (sub)layers. The same is true for the RING-network, where additionally encryption between transport layer entities (virtual-link-by-virtual-link encryption) would be needed to simulate the physical unobservability of lines.
Thus, the MIX-, DC- and even the RING-network can be considered to be virtual concepts, i.e. concepts which may be embedded in a wide range of layers. But an efficient implementation has to avoid unnecessary complex layering and reimplementation of functions. Therefore, figure 5 represents the reasonable design which may be transformed into a reasonable implementation along the lines contained in the computer network literature, e.g. [Tane_81].
As pointed out in [DiHe_79 p. 420], the implementation of our design must not introduce features which enable an attacker to distinguish between alternatives which are indistinguishable in our abstract design. Examples of such bad implementations would be the modulation of the output of a MIX corresponding to the random bit strings contained in the messages it received, to output the result of an adder mod 2 where $1+1 \neq 0+0$ and $1+0 \neq 0+1$ (if the analog signal characteristics are considered) directly in the case of the DC-network or pattern sensitive timing jitter in the case of the RING-network. We will consider these flaws in greater depth in section 2 at those places where we will remark on implementations of the concepts for anonymous networks.

2 Performance

The two main performance characteristics of networks are throughput and transfer delay. Their importance depends on the services the network is supposed to offer. Throughput and transfer delay are less critical for services like electronic mail, only throughput is critical for services like file transfer, only transfer delay for services like telephony, and both are critical for video telephony.

2.1 Channel switching

So far, among the anonymous networks only RING-networks based on slotted rings with exhaustive service are suitable for services that require a continuous stream of information with short transfer delay (channel switching) because once a station is allowed to use a slot, it can use this slot again and again as a channel.
The token ring would also allow simplex channels because of exhaustive service (which was necessary for anonymity), but then most of the time the whole ring would be blocked by just one station. Of course, if we allow slotted rings with only very few slots or rings where the token protocol is executed on many different channels independently, the two concepts become very similar.

The MIX-network is inappropriate for such services because of the delay during the transport of each message. The DC-network described so far is inappropriate as well, because the basic medium access protocols do not guarantee synchronous service (called isochronous in [Sze_85 p. 819]).
New possibilities of increasing the performance of these networks are obtained by dropping one requirement for anonymity that seems unreasonable for services requiring a continuous stream of information with short transfer delay anyway: the requirement that the relationship between different messages of the same connection is hidden.

2.1.1 RING-network and DC-network

In RING- and in DC-networks as in all networks implementing a broadcast channel, channels can be switched by time division. This is discussed in detail in section 2.2.1 and section 2.2.2.

2.1.2 MIX-network

In the MIX-network in its pure form (as described in [Chau_81] and section 1.3), the delay results essentially from the fact that every MIX has to await all bits of a long block before it can decrypt it and send the first bit to the next MIX.
This can be avoided if a single message is used for setting up a connection (this can be a virtual circuit [Tane_81 p. 188] if bursty traffic requiring short transfer delay is to be handled or a channel in case of a continuous stream) and giving each MIX a key of a fast private key (= secret key = conventional = symmetric = one-key) cryptosystem used as a stream cipher. These keys replace the random bit strings in the message, so it looks somewhat like a return address, but it is addressed to the recipient and contains the message content that a connection has been set up.
These private keys are used by the sender to encrypt and by the MIXes to decrypt the following bits of the initiated connection just like the public keys are used by the sender to encrypt and the corresponding private keys to decrypt in the normal MIX-network, but no random bit strings need to be added to the following bits, because the keys are secret (like in the reply message in the return address scheme). Between adjacent MIXes usual connections must be switched on the lower layers, so that they recognize which bits belong to this connection of the higher layer and that they can guarantee synchronous service to the sender and recipient.
Besides shorter delay, this can save ciphering and addressing expense.

To hide the relation between the channels to and from a

particular MIX, the MIX has to await at least two requests for channels of equal bandwidth or it has to establish at least one dummy channel. Otherwise an attacker could infer that the newly established channel (of bandwidth b) to the MIX leaves the MIX as the newly established channel (of bandwidth b) from the MIX.
The same problem is encountered when a channel is to be released. All (or at least two) channels of equal bandwidth which were established at the same time should be released simultaneously. This will cause severe overhead if the mean variation of channel usage is high.
Please note that this problem is not introduced by channel switching: if packet switching is employed for services requiring a continuous stream of information with short transfer delay, the same can be infered by a statistical analysis of the packet rates between sender (as well as recipient) and MIXes and even between MIXes, if all packets use the same route.

After pointing out that channels have to be established and released simultaneously, we have to sketch appropriate implementations. First, we will consider simplex channels, and thereafter duplex ones.

We begin by showing the set-up-scheme more formally. When station S (sender) wants to use a simplex channel through the MIXes MIX_1, MIX_2, ..., MIX_n to station A (addressee), it sends the message

$$A_1, e_1(C, k_1, A_2, e_2(C, k_2, A_3, e_3 (\ldots A_{n+1}, e_n(C, k_n) \ldots))),$$

where e_i is the public key of MIX_i, $e_i(X)$ means X enciphered using e_i; k_i is the key of the cryptosystem used as stream cipher MIX_i shall use, A_i is the address of MIX_i, and A_{n+1} the address of A. C means "Please set up a channel".
Of course, if a broadcast medium is used between MIXes (e.g. a communication satellite), invisible addresses can be used instead of visible ones. The corresponding message may look like

$$e_1(Y, C, k_1, e_2(Y, C, k_2, e_3 (\ldots e_n(Y, C, k_n) \ldots)))$$

where Y is a special bit combination meaning "It's addressed to you".
Thereafter, S starts sending its bitstream bs continuously

encoded as

$$k_1(k_2(k_3(\ldots k_n(bs) \ldots)))$$

where $k_i(X)$ denotes X enciphered using the private key cryptosystem with key k_i as stream cipher.

If the sender starts sending a meaningful bitstream immediately after the set up message, and a MIX cannot switch a channel immediately (which will be the case most of the time, because it must wait for other set up wishes), it has to buffer the bitstream. This is expensive and increases transmission delay, so it should be avoided. But the bitstreams on channels that are mixed together by some MIX must all start at once after the connection set up (or at least at exactly the same time, which would be difficult to achieve in any other way). If the stream cipher is self-synchronous [Denn_82 p. 136, 144], we can use the following scheme:
S starts sending encrypted garbage for an appropriate time and starts its meaningful bitstream with, let's say, one hundred ones followed by a zero (since that sequence is known to an attacker, the cryptosystem used as stream cipher must withstand a known plaintext attack as does any good cryptosystem). The MIXes start deciphering the incoming bitstream when it arrives and transmit the deciphered bitstream without buffering to the next one as soon as a channel can be switched. If once the channel cannot be switched fast enough, so that the receiver misses the beginning of the meaningful part of the bitstream, this can be treated with end-to-end fault tolerance mechanisms (see section 3.1).
Alternatively we could try to inform the sender when the channel has been established, so that he only changes from garbage to a meaningful bitstream thereafter.

To release the channel, S sends, let's say, a one followed by one hundred zeros to A, stops encryption and sends a one followed by one hundred zeros to the first MIX and then stops sending. After reception of a one and one hundred zeros in a row (which is, since stream encryption is used, much less likely to happen by chance then being hit by a truck, an SS-20, or

Pershing, in which cases communication as well as privacy becomes useless for the victim) A knows that the incoming bits are garbage. After reception of its one and one hundred zeros in a row the MIX knows that the channel can be released, but it continues to send garbage on it until it can release at least two simultaneously established channels of equal bandwidth. To do that, it stops decryption and sends a one followed by one hundred zeros on each channel which is to be released and then stops sending on them. This continues, until the addressee receives a one followed by one hundred zeros a second time.
Since each MIX has complete control which of the simultaneously established channels it releases simultaneously, all channels can be released. Since the release of channels can mutually block only if they are established simultaneously, chronological ordering prevents deadlocks. More precisely, a part of a channnel can be released at the latest when the senders of all channels of which the connection set up wish had been uttered before this part of the channel was switched have declared that those channels can be released.

To establish a duplex channel, we have two possibilities:

1) First establish a simplex channel from the connection initiator (sender 1) to the called party (addressee 1) and thereafter a simplex channel through possibly different MIXes from the called party (sender 2) to the connection initiator (addressee 2), e.g. using an untraceable return address.
   The first disadvantage of this possibility is that the time to set up a channel is needed twice before duplex communication is possible.
   The second disadvantage is that the causual connection between the two simplex channels and its observable manifestation (chronological correlation) can undermine anonymity.
2) Establish a duplex channel between S and MIX_1, thereafter between MIX_1 and MIX_2, ..., thereafter between MIX_n and A. The same kind of message may be used for connection setup of duplex channels as for simplex channels.

In this case, not only the sender but also the MIX starts sending encrypted garbage at once. If the other rules concerning establishment and release discussed in the context of simplex channels are respected, anonymity is created. Note that even in case of duplex channels, where the connection initiator or the called party can terminate a connection, channels can be released from connection initiator through the MIXes to the called party successively. When the called party terminates the connection, this is signaled by his station to the station of the connection initiator, which then releases the channel.

2.2 Some remarks on implementations of the basic concepts for anonymous networks

Analyzing the performance of the concepts of section 1 must be concurrent with considering how they would be implemented physically.
First, we sketch a cost-efficient implementation of the RING-network and compare its performance with that of a star-network.
Next, we describe how the DC-network can be implemented cost-efficiently and which anonymous medium access protocols manage its broadcast channel efficiently.
Finally, we consider how many stations may act as MIXes in a MIX-network and how many MIXes a message may pass. To answer these questions we develop a simple model to study performance and unobservability.

2.2.1 RING-network

Since the sole ring network standard, the token ring standard, prescribes a "Source Address" and a "maximum period of time the DTE may transmit frames after capturing a token" [ECMA89_85 p. 8, 20] a RING-network cannot conform to it if implemented as depicted in figure 5.

Therefore, we have to sketch a suitable implementation. Afterwards we will compare its performance with that of an equally expensive star-network operated on a "transmission on demand basis" as planned by the german PTT.

2.2.1.1 Cost-efficient implementation

A cost-efficient implementation of the RING-network has to

1) keep the delay in each station very small and the lines as short as possible, since the worst case delay of the RING-network is the sum of the delays of all stations (except the sending one) and the delays of all lines (except the incoming line of the sending station),
2) provide anonymous communication to the data link layer as substantiated in section 1.5 and depicted in figure 5,
3) keep the number of gates operating at the ring data rate (or even above) to a minimum, since these gates are not as cheap as slower ones and consume more power, and
4) minimize the amount of traffic caused by each connection.

The first three points are mainly concerned with medium and physical layer of the OSI model, whereas the fourth is concerned with the data link layer. But let us now consider these points in turn.

1) The delay in each station is the quotient of the bit delay in the station (must be ≥ 1) and the ring data rate. Therefore, at very high ring data rates we can afford some bits delay per station.
   The delay of the lines is approximately independent of the ring data rate [AlFi_77 p. 572, 573]. Therefore, all we can do is to keep the lines as short as possible. This requirement (and the requirement for physical unobservability, cf. section 1.4.2, as well) is satisfied by connecting the stations directly. To give an example, in a house divided into several flats the ring connects the stations in the flats directly and not via a central wiring concentrator.
2) Analogue signals are transmitted on the medium. Digital

signal (re)generation recovers amplitude, shape, and timing of these signals. This must be done so that an attacker knowing the physical characteristics of all stations exactly cannot deduce any information from it.

To motivate this requirement, we give an example of a well known ring implementation which does not satisfy this requirement and therefore allows an attacker to identify the sender of a message:

As described in [BCKK_83, KeMM_83, BaSa_85], the token ring implemented at the IBM Laboratory in Zurich generates bit pattern sensitive timing jitter in each station, as does any implementation of the token ring standard [ECMA89_85 chapter 6.4 and 6.5]. The accumulated jitter is compensated by the use of an elastic buffer and a master clock in one station executing the monitor function in the ring. An attacker surrounding two stations, where neither of them executes the monitor function, records the bit pattern and the exact timing of the first station's input and the bit pattern and the exact timing of the second station's output. If the bit patterns are different (in which case the attacker should be uncertain which of the two patterns was transmitted between the two stations) the attacker calculates the resulting timing of both possible bit patterns between the two stations and compares this with the timing he observed. If we ignore noise between the two stations, our attacker can deterministically decide whether the first or the second station sent. If there is only marginal noise, which is the aim of every network design, the attacker can decide with high probability which station sent.

To show that the requirement can be fulfilled, we sketch three appropriate implementations:

* Each station uses its quartz oscillator for transmission and derives timing information for receiving with a phase-locked loop. The frequency differences between stations are compensated by inserting or deleting fill bits in the data stream. This is accomplished with an elastic buffer in each station holding a few bits. Bits

will be inserted or deleted when the buffer tends to underflow or overflow, respectively [KeMM_83 p. 724].

* Instead of inserting or deleting bits, the frequency of the local quartz oscillator is changed very slowly depending on the actual filling of the elastic buffer [KeMM_83 p. 725].
* Each station receives timing information from a network reference clock, as planned by the PTT anyway [McLi_85 p. 341], and uses this timing information to control the frequency of its local quartz oscillator.

3) To provide the required bandwidth, monomode optical fibers must be used as links. To keep the number of gates operating at the ring data rate (or even above) small, we should divide the bandwidth of each link into a couple of channels. This can be achieved by Wavelength Division Multiplexing (WDM) [Unge_84 p. 154], which is especially appropriate if otherwise the limit of electronic pulse-forming and driving circuits would be reached or exceeded.
If this limit is not reached, coherent detection [BaBr_85, Stan_85] or Time Division Multiplexing (TDM) [BeEn_85, GDRB_85] is appropriate. An operational implementation of ring stations operating with 5 Gbit/s using TDM is reported in [BeEn_85].
Each multiplexed channel can be managed by an anonymous ring access protocol or some channels can be managed by a signalling protocol executed in another channel. As explained in [Pfi1_83 p. 61], this has the advantage that each station needs only logic for address recognition for a few channels. A station only listens to some of the other channels after executing a protocol for anonymous channel switching. If the bandwidth of these other channels is chosen appropriately, the bandwidth can be used fully and therefore no addressing is needed on these channels.
To adapt the number and bandwidth of channels at higher layers to the ever changing communication demand, there the partitioning of the ring bandwidth into channels can be done dynamically, especially if TDM is used [GöKü_85, Göld_85].

4) In case of duplex channels, the bandwidth required per

connection can be halved if the medium access protocol slotted ring is used: both partners fill each slot used by the connection whenever one arrives.
This modification reduces anonymity only slightly: If an attacker cannot determine that a slot is used twice per round, anonymity is not reduced at all. Otherwise, an attacker surrounding two stations which establish a duplex channel between each other can observe that. In all other cases the proof sketched in figure 4 holds [Höck_85 p. 53].
If desired, stations may acknowledge the correct reception of the data by setting a bit, since the content of a slot does not return to its sender, so it cannot check for transmission errors itself.

2.2.1.2 Performance comparison of star-network and RING-network

In local areas with a few hundred stations the performance of a RING-network implemented as a physical ring is roughly as good as, or even better than, that of an equally expensive usual star network operated on a "transmission on demand" basis or that of a broadcast bus network [Pfi1_83, Bürl_84, Bürl_85, Mann_85].
The first comparison result is a big surprise to most people (provided that they believe it at all). It stems from the facts that:

* Rings of N stations require only N senders and N receivers (which is the minimum for any network permitting two way communication), whereas stars require 2N of each. 2 senders or receivers cost roughly twice as much as one, whereas one sender or receiver with twice the price will provide considerably more than twice the bandwidth, if we keep distance from the limits of technology.
* The overall cable length of a ring grows proportionally to the square root of the number of stations in a given area, whereas the overall cable length of a star grows proportionally to the number of stations. The overall length of cable ducts is essentially the same for star- and RING-networks.
* As far as packet and message switching is concerned, the

shared medium of a RING-network is a "wider" bottleneck than the shared complex switching center of a usual star network operated on a "transmission on demand" basis.

* In the foreseeable future, the bulk of traffic in any ISDN will be mass communication: not only one station but a significant fraction of all stations will receive it. For this traffic, a broadcast network is an economical choice. In the far future, bandwidth will be (approximately) for free anyway.

Therefore, even people absolutely not concerned about user observability propose connected ring networks as Metropolitan Area Network [Sze_85].

However, for the foreseeable future, performance and reliability or cost of large RING-networks, e.g. with more than 10000 stations, become unacceptable for a broadband ISDN.

2.2.2 DC-network

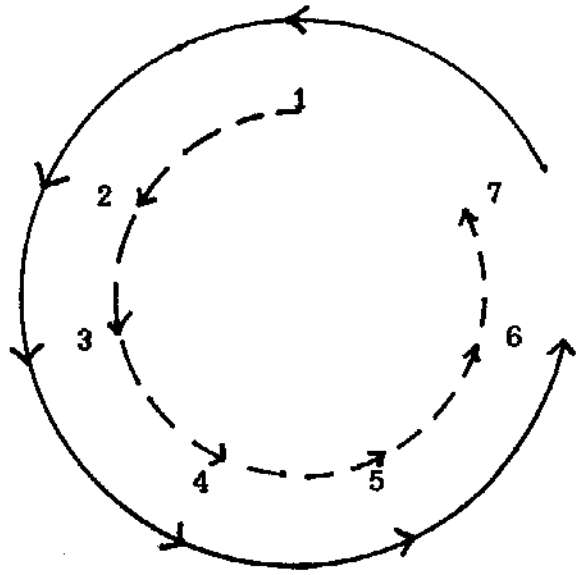
In the first subsection we are mainly concerned with medium and physical layer of the OSI model, i.e. we will consider how the DC-network can be implemented cost-efficiently.
In the second subsection we are concerned with the data link layer of the OSI model, i.e we will consider anonymous medium access protocols which can manage the anonymous broadcast channel more efficiently than those cited in section 1.4.1.

2.2.2.1 Cost-efficient implementation

First, we will study topologies suitable for the implementation of the DC-network.
Afterwards, we will explain how the number of required high speed gates can be minimized by division of the broadcast channel into subchannels.

2.2.2.1.1 Suitable topologies

In [Cha3_85], David Chaum suggests to implement superposed sending on a physical ring network. Each message bit requires (nearly) two circulations around the ring: in the first round, the user bits are successively superposed by the users, in the second round, the resulting bit is broadcasted. "Nearly" means, that in a ring of N stations we only need 2N - 2 transmissions from station to station, since the last station in the first round receives the outcome already then. So it need not receive it again in the second round (see figure 6).



Station 1 starts sending its bit.
Stations 2 to 7 superpose their bits.
Station 7 knows the outcome which is transmitted to the stations 1 to 6.

— — → transmission for superposition
——→ transmission for broadcast

Fig. 6: Exact number of circulations in the DC-network implemented on a physical ring network

This implementation seems quite efficient because under the assumption of uniformly distributed traffic it increases the average expenditure of transmission only by a factor of slightly less than four compared with a traditional ring access protocol in which the recipient removes the message from the ring, whereas on a star or tree network the factor is the number of stations. But the amount of transmission on each line, i.e. the

required bandwidth, is the same for all implementations, so implementations on stars or trees might still be better if their round trip transfer delay is shorter. We will now show that this is indeed possible.

Naturally, the transfer delay of the network is the sum of the transfer delay caused by switching and the transfer delay caused by transmission.
We will see that

1a) the transfer delay caused by switching in a star or tree network need only be proportional to the logarithm of the number N of stations, whereas

1b) in ring networks it is always proportional to N, and

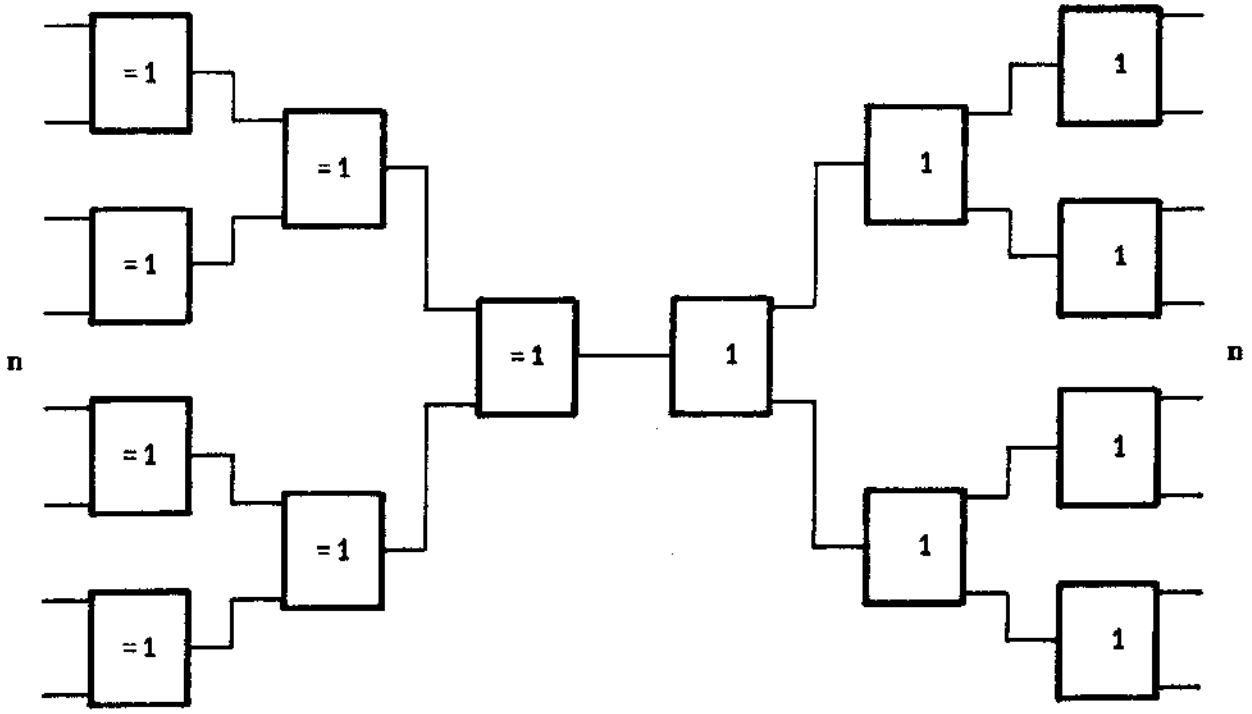
2a) the transfer delay caused by transmission in a star or tree network is proportional to the product of the distance d of neighbour stations and something between the third root of N and N itself (the square root may be a typical value), whereas

2b) in ring networks it is always proportional to that product of d and N.

An appropriate node ("switching center") for a tree or star network for 1a) is shown in figure 7. It can be seen that it is much less complex than a normal switching center.

tree of XOR-gates for adding:
delay caused by switching
proportional to ld n

tree of amplifiers:
delay caused by switching
proportional to ld n



Each of the inputs on the left is connected to the output of one station,

each of the outputs on the right is connected to the input of one station.

Additional circuitry for timing is ignored.

Fig. 7: Node ("switching" center) for the DC-network

1b) stems from the fact that each bit can only be passed on after it has been received completely, so that each station causes at least one bit time delay.

The transfer delay caused by transmission is proportional to the diameter of the network, which is the maximum over all ordered pairs of their minimal distance (geodesic [Tane_81 p. 37]) in the corresponding weighted directed graph.
If there is at most one user station in each cube with edges of length d ($d \geq 1m$ seems quite reasonable), the diameter of any network with N stations, even if it is fully connected, is at

least the product of d and the third root of N. If our cubes are arranged in a plane, which seems to be an appropriate model of the surface of the earth and therefore appropriate for rural areas, the diameter of any network in rural areas is proportional to the product of d and the square root of N. Since appropriately designed hierarchical networks have a diameter close to that of a fully connected network, the square root seems to be a typical value. Of course, if all user stations are arranged along a straight line, the best diameter we can achieve is the product of d and N. This yields 2a).
With the same assumptions, the diameter of a ring network is at least the product of N-1 and d, which yields 2b).

So if we fix a minimum distance and describe the influence of an enlargement of our network in the sense that new user stations are connected, thereby increasing the geographical span of the network, the transfer delay caused by transmission is predominant. So all we can achieve in this case by the construction of a network with suitable topology is an improvement of the overall network delay by a factor which is proportional to the square root of n.
If, on the other hand, we connect new user stations without increasing the geographical span of the network, which will be the case if e.g. the span of the network is given by the frontiers of a particular country in which more and more citizens realize that observability is a severe threat and join our DC-network, we can achieve a very significant improvement of the overall network delay. The network delay only increases with the logarithm of the number of users, whereas the overall delay of the ring implementation grows with the number of users.
For reasonable finite numbers of stations and very high bit rates, (which are typical for a broadband ISDN) the transfer delay caused by switching is less significant than the transfer delay caused by transmission, e.g. if our network operates at 5 Gbit/s, light only travels 4 cm in an optical fiber in the time needed to transfer one bit, so here again the overall network delay is shorter by a factor of about the square root of N on a suitable star or tree topology than on a ring.

2.2.2.1.2 Minimization of the number of required high speed gates

As we have explained in section 2.2.1.1 3) the use of high speed gates should be economical by the division of the broadcast channel into subchannels.
A couple of channels can be multiplexed over monomode optical fibers using WDM, coherent detection, or TDM. If these channels should use the same XOR-gates, amplifiers and timing circuitry depends on their speed, cost, and reliability. If the overall cost of the XOR-gates, amplifiers, and timing circuits is not much bigger (or even less) when they are dedicated to single channels we should choose this solution, because it provides graceful degradation, i.e. the failure of an XOR-gate, amplifier or timing circuit only affects the corresponding channel.
The same thoughts apply to pseudorandom key generators and the gates which superpose message and keys inside the stations. Please note that the superposition of keys and message must be implemented in a way that no characteristics of the signal leaving the station give hints which value the keys and the message had. This corresponds to the problem described in section 2.2.1.1 2).
Each multiplexed channel can be managed by an anonymous medium access protocol or some channels can be managed by a signalling protocol executed in another channel. As explained in section 2.2.1.1 3), this has the advantage that each station needs only logic for address recognition for a few channels.

As throughput and cost (as well as reliability, but not necessarily availability, cf. section 3) of any network based on superposed sending cannot be superior to that of a RING-network, in the foreseeable future these networks cannot be built with more than 10000 stations either.

2.2.2.2 Efficient anonymous medium access protocols

First, we will mention that slotted ALOHA can easily be extended (called R-ALOHA in [Tasa_83]) to allow the transmission of a continuous stream of information in nearly the same fashion as in RING-networks based on slotted rings with exhaustive service (cf. section 2.1):

The time is divided into slots, which are organized into frames. The duration of a frame is chosen to be greater than the round trip propagation delay of the DC-network. Consequently, each station is aware of the usage status of slots one frame ago.

A station which successfully transmits in slot m (say), has the exclusive right to use it in the next frame.

If an end-of-use bit is included in each slot, it need not be empty to signal that another station may try to use it.

Secondly, we will explain how the anonymous reservation scheme mentioned in [Cha3_85] can be extended to avoid any collisions (a figure describing the achievable channel utilization of reservation schemes as a function of the ratio

request bandwidth / message bandwidth

and comparing it with that of slotted ALOHA and CSMA can be found in [Toba_80 p. 479]):

Instead of using single bits for reservations as described, use entities of m bits (with 2^m greater than the number of stations in the DC-network). Instead of inverting one bit to signal a reservation (which has the same outcome, whether one, three, five, ... stations do it), add 1 to one of the entities of m bits interpreted as binary integer. Of course, this interpretation has to be maintained through the whole network during the reservation phase and also the keys must be interpreted as m-bit binary integers. Note that instead of adding bits modulo 2, m bits may be added modulo 2^m. In this case, one and only one of the two parties knowing each key has to subtract it, so that superposed sending still works. (This can be the extension of superposed sending to other fields than GF(2) that David Chaum mentions in [Cha3_85] without

giving any application for it, though more generally we do not take a field but $\mathbb{Z}/2^m\mathbb{Z}$ and could take any abelian group.) If the final value of an entity of m bits is zero, it was not used to signal a reservation. If the value is one, a reservation was signalled successfully. Otherwise, several reservations were signalled unsuccessfully.

Thirdly, we will explain how an Announced Retransmission Random Access (ARRA) protocol [Rayc_85] can be adopted. The version under consideration (extended ARRA) has the advantages that, if the load is light, most packets can be transmitted with one round trip propagation delay (whereas in any reservation scheme, at least two round trip propagation delays are required), and if a packet suffers from a collision, the retransmission announcement prevents further collisions of that packet (further collisions can happen e.g. in slotted ALOHA).

Use extended ARRA exactly as described in [Rayc_85] and implement the minislots using the above described entities of m bits, implement the announcement of a retransmission by a minislot by addition of 1 to the m bits interpreted as binary integer and implement the ternary feedback of the message slots by, e.g. a CRC-character.

If you do not want to maintain entities of m bits through the whole network, use a probabilistic scheme, e.g. invert an arbitrary one of the m bits.

Fourthly, we have to mention that the two collision-resolution algorithms described in [Mass_81] can be used instead of ARRA to push the achievable throughput beyond 1/e, which is the limit of slotted ALOHA.

Please note that slotted ALOHA and both the third and the fourth medium access protocol give an attacker the information that the packets which suffered the same collision are not sent by the same station.

In the case of slotted ALOHA an attacker can calculate the superposition of all slots in a certain time window. If the superposition of some slots is exactly the content of another

slot sent some time ago, the probability is high that all these slots suffered the same collision and therefore are sent by distinct parties. Of course, this can be hidden by changing the encoding of all messages which must be retransmitted after a collision.
In case of the third and fourth medium access protocol, the medium access protocol itself gives the information which packets suffered the same collision to all stations of the DC-network. Therefore, changing the encoding of messages does not help. The only possibility in this case (which helps in the previous case as well) is that a station can simulate collisions by its own packets, but this of course decreases throughput.

Next, we will explain how carrier sensing (CSMA) [Tane_81 p. 288] can be used to improve the achievable utilization beyond 1/e, which is reachable with slotted ALOHA, without incurring at least two round trip propagation delays as does any reservation scheme and without establishing relations between packets, e.g. that they are not sent by the same station.
Of course, carrier sensing can only help if the round trip propagation delay of the network is short compared to the packet or message transmission time [Toba_80 p. 474] (the comparison of [KuSY_84 p. 57] does not apply since in the DC-network a message does not have to propagate through the whole network before the channel may be reused). Since the lengths of messages and (to a lesser degree) suitable lengths of packets are determined by the use of the network and therefore are not at the discretion of the network designer, all that can be done is to build DC-networks with short round trip propagation delay (cf. section 2.2.2.1) and to use TDM to increase the packet and message transmission time, where appropriate (but note that TDM incurs delay which we wanted to reduce compared with reservation schemes).
To avoid that a station sensing the carrier can be identified by the time of its behaviour, all stations should sense the bits on the carrier at the same time (of course this does not necessarily mean that all bits arrive at all stations at the same physical time; it suffices that the output of bit i of all stations only

takes into account all bits until bit i-j, where j is chosen such that all stations receive bit i-j in good time).
Then persistent as well as nonpersistent CSMA may be used.
Since all stations react absolutely equal, even collision detection (CSMA/CD) may be employed: a station which receives as bit i-j another value than it sent, ceases sending its message at bit i. This completely digital collision detection (in contrast to, e.g. Ethernet, where analog collision detection is employed [Tane_81 p. 293]) guarantees that any sending station which got its message garbled notices this (and may retransmit its message later on). Furthermore, completely digital collision detection makes it possible that one message is not changed at all by the collision, so the station sending it does not notice the collision at all and successfully transmits its message. Let's be happy about it! End-to-end encryption working (among other things) as an error detecting code will catch truncated messages. So let's beware of jamming after a collision has been detected!
If j must be chosen to be great and an attacker must not learn that the messages involved in a collision are not from the same station, the encoding of messages which suffered from a collision where all stations aborted their message transmission should be changed.

To sum up, the medium access protocols used to efficiently manage the broadcast channel of the DC-network must preserve anonymity, as noted in section 1.5.
In the proof of anonymity of the DC-network [Cha3_85] all stations are equal to each other. This is in contrast to the RING-network, where there are special relationships between stations, e.g. to be neighbor stations, which in a certain sense limit anonymity and make the proof of this (limited) anonymity more complicated but enable more efficient use of the bandwidth by distributed and unobservable polling, whereas between stations which are equal to each other collisions are always possible. If reservations are used to avoid collisions of packets, the reservations can collide. If reservation-reserva-

tions are used to avoid collisions of reservations, the reservation-reservations can collide, and so forth.
As an informal criterion that may be used to judge other published or forthcoming medium access protocols for suitability to manage the bandwidth of the DC-network, we state that all medium access protocols which

* treat all stations as absolutely equal (e.g. stations have no unique number as in the Bit-Map Protocol of [Tane_81 p. 296]),
* do not establish relationships between messages (e.g. "if this message is sent by station A, that message is sent by station B"; where A=B occurs in practice more often than not), and
* allow a station to use the whole bandwidth of the broadcast channel if the other stations do not send anything at all

preserve anonymity completely.

Of course, a formal treatment of the anonymity of single protocols as well as of the above stated criterion as well as of anonymity preservation in layered systems in general seems worthwhile.

2.2.3 MIX-network

In the MIX-network, several factors are to be considered: How many and which stations act as MIXes and how many MIXes are used per message?

The message length grows at least proportionally with the number of MIXes chosen. Here message means any independent entity, not e.g. a part of the information on a channel. (If we have a starting message and then several others depending on it whose length does not grow, asymptotically it does not matter whether we consider only the starting message or the total.) Especially this growth exists for the pure scheme, the return address scheme (here it is the address part which grows) and also for the shortly mentioned scheme where messages keep their length when they are mixed. The reason is the inclusion of at least 100

random bits for each MIX. This was necessary because the public key of the MIX must be used for encryption of at least a part of the message, because MIX and sender have no other key in common, and otherwise an attacker could try to reencrypt the output messages of a MIX and see to which input messages they correspond.
If we use the pure scheme in a natural way, the message length grows even exponentially, because the cryptosystem has a fixed block length, e.g. 1000 bits for RSA, and we need 100 random bits for each block as independently encrypted entity. Therefore, to derive the message M_i, which MIX_i shall receive, from M_{i+1}, the sender breaks M_{i+1} into blocks of 900 bits, concatenates 100 random bits to each and encrypts it with e_i. So M_i is at least 10/9 times larger than M_{i+1}. The same holds for the address part of the return address scheme.
Luckily, this is not the case for the shortly mentioned scheme where messages keep their length, nor for the fault tolerance schemes which will be described in 3.1.2.3.1, because in these schemes each block of the message is only encrypted with a public key cryptosystem once and the other times with a private key cryptosystem (and there is essentially one block per MIX).

In any case this implies that the expenditure of transmission of a message grows at least quadratically with the number of MIXes chosen for it (as the messages are both longer and transmitted more times). So this number must not be too large. Especially, not all stations can be chosen as MIXes for all messages in large networks.
Fortunately, the contents of established channels do not grow with the number of MIXes chosen, since we may use a conventional stream cipher. But since the delay of each channel (and its expense as well) is proportional to the number of MIXes chosen, this number cannot be too large in this case either.

To guarantee short transfer delay for time critical services, the throughput of a station that acts as a MIX must be very high because it must always have enough messages to mix. These messages must be decrypted, rearranged, and forwarded. Thus, a

MIX should be extremely powerful and complex, and therefore will not be cheap. Consequently, only a limited number of MIXes can be afforded in the network.
If the MIX-network is implemented using some user stations of an existing physical network as MIXes, each message must pass through the physical network several times, which adds additional delay to that occurring in the MIXes. But using the switching centers of the physical network as MIXes cannot be recommended either, because the probability that they collude is too great (and the assumption that they are independent becomes altogether absurd in countries with a telecommunication monopoly as in the FRG).

To get a better understanding of the interrelation of design choices, let us consider the following model of our world. We first describe the system under consideration, then its load and our assumptions. Thereafter, some performance formulas are given, which lead to upper bounds on the number of MIXes that can be used per message or channel and the total number of MIXes in the network. The formulas are applied to some performance scenarios. Along their lines, other performance scenarios can be developed easily.

Notations concerning the system

Let be

N	the number of users,
M	the number of MIXes, where $M \leq N$,
U [bit]	the smallest encrypted unit of the cryptosystem used,
	e.g. U = 1000 for the RSA public key cryptosystem
	or $1 \leq U \leq 64$ for the DES used as a stream cipher,
T [sec]	the time to encrypt/decrypt a unit of the cryptosystem,
	e.g. T = 1/64 for the RSA public key cryptosystem
	or 1 nsec $\leq T \leq$ 4.5 microsec for a stream cipher
	(the latter value is calculated by the performance
	of the fastest DES chip: 64 bit / 14000000 bit per sec)
D_{MtM} [sec]	the MIX-to-MIX delay, i.e. the time per MIX for
	switching and transmission without decryption,

e.g. 0.001 sec = 200 km / 200000 km per sec,
that means, MIXes are at most 200 km apart

D_{tra}[sec] the maximal transmission delay in the network without any MIXes (one direction) =
network diameter / propagation delay,
e.g. in a worldwide network
D_{tra} = 20037 km / 200000 km per sec ≈ 0.1 sec

Notations and assumptions concerning the load

We assume

* equally distributed traffic between users,
* equally distributed traffic through MIXes
  (otherwise the situation for MIXes with less traffic and so the situation for users using these MIXes gets worse),
* that the same number of MIXes are used per connection, message, or packet,
* that, if dummy-traffic is used at all, it is used end-to-end, so that dummy messages can be treated as increasing the rate of normal messages, and
* that each MIX outputs batches of messages or packets or set up connections, respectively, at fixed time intervals.

So let be

r_{ev}[sec^{-1}] the rate of the event under consideration, e.g. that
a user initiates a connection ($0.01 \leq r_{ev} \leq 10^{-6}$),
a user sends a message (typical r_{ev} = 0.0001) or
a user sends a packet (typical r_{ev} = 0.01)

a [sec] the acceptable delay, e.g.
connection set up for (video) telephony: 10 sec,
transfer delay (one direction) for
(video) telephony: 0.2 sec
interactive videotex: 10 sec
electronic mail: 1000 sec

r_{tra}[bit/sec] the bit rate of transmission in the network

r_{sen}[bit/sec] the bit rate of the service at the sender, e.g.
telephony: 64 kbit/sec
video telephony: 34 Mbit/sec

u [bit] the frame length of the service, i.e. the number of

message bits in each encrypted unit. We have $u \leq U$ and typically $u=U$ for a stream cipher and $u=U-100$ for the pure MIX scheme because of the random bits. Choosing a smaller u can shorten the delay, but at the cost of greater transmission overhead.

m — the number of MIXes used per message

t [sec] — the length of the interval between two (batch) outputs of a MIX

We will further assume that in case of packet switching the packet length divides U, so that there is no additional delay for reblocking. In reality these packet lengths can be rather great (e.g. at least 80·20·8 bit = 12800 bit for interactive videotex or about 100000 bit for electronic mail). This would prevent them from dividing U.
We will also ignore the growth of the message due to inclusion of random bits in case the original MIX-scheme with a public key cryptosystem is used.
(These are conservative assumptions if we derive upper bounds for the numbers m and M.)

Formulas

We want to compute

v, the (expected) number of events (see r_{ev}) per MIX and batch,

D [sec], the maximal delay of messages,

and then our minimum requirements are

(1) $v \geq 2$

(2) $D \leq a$.

We have

$$v = \frac{N \cdot r_{ev} \cdot t \cdot m}{M} .$$

D can be computed as

$$D = D_{sen} + D_{tra} + D_{MIX},$$

where D_{sen} denotes the (maximal) delay at the sender, and D_{MIX} that by the m MIXes. D_{tra} is the maximum transmission time for messages in the network without any MIXes. (If every user chose his m independently, those who knew where the recipient lived could work with the real transmission time instead of D_{tra}.)

We have

$$D_{sen} = \frac{u}{r_{sen}} + m \cdot T$$

because u bits must be awaited and then encrypted m times.
D_{MIX} depends on how the times are coordinated when the different MIXes output batches. In any case

$$D_{MIX} \geq m \cdot (D_{MtM} + D_{cry}),$$

where D_{cry} denotes the delay caused by decryption at each MIX:

$$D_{cry} = \frac{U}{r_{tra}} + T .$$

If the MIXes are completely uncoordinated, we must take

$$D_{MIX} = m \cdot (D_{MtM} + D_{cry} + t),$$

because at each MIX it can happen that the message arrives just too late to be included in one batch and thus has to wait for nearly the time t in addition to the time for mixing. If we were satisfied if the delay is acceptable with high probability, we could take

$$D_{MIX} = m \cdot (D_{MtM} + D_{cry} + c \cdot t)$$

for some c with $0.5<c<1$, because $0.5 \cdot t$ is the expected waiting time at each MIX.
If all MIXes output their batches at the same time, we must take at least

$$D_{MIX} = m \cdot \max \{D_{MtM} + D_{cry}, t\}$$

(If $t < D_{MtM} + D_{cry}$, then more precisely D_{MIX} is m times the smallest multiple of t larger than $D_{MtM} + D_{cry}$, but making t smaller than $D_{MtM} + D_{cry}$ makes no sense in this case, because the messages coming from other MIXes are not ready to be output at that time yet. Also more precisely at the first MIX the message may have to wait the complete time t.)
As long as the users may choose arbitrary sequences of MIXes, we cannot do much better than that. For $D_{MtM}+D_{cry} < (M-1)/M \cdot t$ the best possibility is that the M MIXes output batches one at a time at intervals of t/M. Then we can guarantee

$$D_{MIX} = m \cdot \frac{M-1}{M} \cdot t = m \cdot \max \{D_{MtM} + D_{cry}, \frac{M-1}{M} \cdot t\}.$$

If we restrict the users' freedom to choose arbitrary sequences

of MIXes, we can do much better for large t by dividing the MIXes into classes and letting each class output its batches $D_{MtM}+D_{cry}$ seconds after the previous one. If the users are forced to choose MIXes of successive classes for their messages, it is guaranteed that the messages must only wait at the first MIX, so

$$D_{MIX} = t + m \cdot (D_{MtM} + D_{cry}).$$

In any case our requirements are

$$(1) \quad M \leq \frac{1}{2} \cdot m \cdot t \cdot r_{ev} \cdot N$$

$$(2) \quad \frac{u}{r_{sen}} + m \cdot T + D_{tra} + D_{MIX} \leq a .$$

For the case of arbitrary sequences of MIXes and simultaneous output of batches we get

$$(2) \Leftrightarrow (2'): \quad m \cdot \max \{D_{MtM} + \frac{U}{r_{tra}} + 2T,\ t+T\} \leq a - D_{tra} - \frac{u}{r_{sen}}$$

and for fixed orders between MIXes

$$(2) \Leftrightarrow (2''): \quad t + m \cdot (D_{MtM} + \frac{U}{r_{tra}} + 2 \cdot T) \leq a - D_{tra} - \frac{u}{r_{sen}} .$$

From now on we will only treat these two cases.
Usually m, t, and M are to be chosen and the rest of the parameters is fixed. For these in our scenarios we take the examples mentioned in the notations.

In all scenarios we will start with some (conservative) estimates to simplify (2') and (2").
E.g. u/r_{sen} is always small compared to a and thus left out. (Also $a - D_{tra}$ will be of the same order of magnitude as a, otherwise a was overspecified.)
Also in most situations two of the three terms in the sum $D_{MtM} + U/r_{tra} + 2 \cdot T$ are small compared with the third and left out, and we will also ignore the term T in t+T in (2').
If we denote the equations (1), (2), (2'), (2") with the special

parameters of a scenario x by (1.x), (2.x), (2'.x), and (2".x), resp., these estimates lead to equations (3'.x) and (3".x) as consequences of (2'.x) and (2".x), resp., of the form

$$(3'.x) \quad m \cdot \max \{ k , t \} \leq b$$

$$(3".x) \quad t + m \cdot k \leq b$$

From both we can derive the same upper bound

$$(4.x) \quad m \leq \frac{b}{k}$$

for the number of MIXes that may be used per message in scenario x.
To derive an upper bound for M, the overall number of MIXes, from (1), we must first derive one for $m \cdot t$ from (3'.x) or (3".x), respectively.
From (3'.x) we simply take $m \cdot t \leq b$, so together with (1) we get an equation

$$(5'.x) \quad M \leq \frac{1}{2} \cdot b \cdot r_{ev} \cdot N$$

(where the values of b and r_{ev} are given).
In the case of restricted orders of MIXes, where we have requirement (3".x), $m \cdot t$ is maximal if $t + m \cdot k = b$, so $m \cdot t = m \cdot (b - k \cdot m)$. This function in m takes its maximum for $m = b/(2 \cdot k)$. So $m \cdot t$ is maximal for $m = b/(2 \cdot k)$ and $t = b - m \cdot k$, thus $t = b/2$. These equations imply

$$m \cdot t \leq \frac{b^2}{4 \cdot k} .$$

Therefore (1) and (3".x) together lead to

$$(5".x) \quad M \leq \frac{1}{2} \cdot \frac{b^2}{4 \cdot k} \cdot r_{ev} \cdot N .$$

Scenario 1: Electronic mail

Here we have a = 1000, so we can ignore D_{tra} and take

$$b = 1000 .$$

Also, because here we have the original MIX scheme with RSA as cryptosystem, D_{MIM} and U/r_{tra} are small compared with $2 \cdot T = 1/32$. So we take

$$k = \frac{1}{32} .$$

This yields

$$(4.1) \quad m \leq 32000$$

$$(5'.1) \quad M \leq \frac{1}{2} \cdot 1000 \cdot 10^{-4} \cdot N = \frac{1}{20} \cdot N$$

$$(5''.1) \quad M \leq \frac{1}{2} \cdot \frac{1000^2 \cdot 32}{4} \cdot 10^{-4} \cdot N = 400 \cdot N$$

So not more than 32000 MIXes can be used per message. If the users can choose arbitrary sequences of MIXes, not more than 5% of the users can act as MIXes. From (5".1) we get a requirement weaker than $M \leq N$, so that in that case all stations can act as MIXes (which was necessary if the MIX scheme was to be used as sender anonymity scheme).

All this looks quite good, but before really using 32000 MIXes per message, one should also consider how to handle the growth of the message length and how complex the MIXes would be (e.g. we only considered a single message, so we somehow assumed that the MIXes mix all messages in parallel).

Now we will concentrate on scenarios with more stringent performance requirements.

Scenario 2: Telephony with circuit switching

Actually, this is a group of scenarios, especially because we have two kinds of events, connection set up and actual transmission. If formulas are specific to one of these two kinds of events, we distinguish them by suffixes 'con' for connection set

up and 'tra' for actual transmission.
For connection set up we must fulfil requirements (1) and (2) as usual, i.e.

(1.2con) enough circuits must be established to gain traffic security, and

(2.2con) the delay of the connection setup must be acceptable.

As we have the original MIX scheme for connection set up again, we take k=1/32 like in scenario 1 and b=a=10 this time. So we have

(4.2con) $m \leq 320$,

(5'.2con) $M \leq \frac{1}{2} \cdot 10^{-3} \cdot N$,

and

(5".2con) $M \leq \frac{1}{2} \cdot \frac{10^2 \cdot 32}{4} \cdot 10^{-4} \cdot N = \frac{1}{25} \cdot N$

This implies that at most 0.05% or 4%, resp., of all stations can act as MIXes.
The requirements for the actual transmission are weaker than usually: If (1.2con) is fulfilled, there is no additional requirement (1) for actual transmission necessary, because a MIX can always mix the corresponding bits of the channels that were set up together as soon as all of them arrived (so we need not fix times when the MIXes output batches a priori), and the bits do not even have to wait at the first MIX once the channel is established (except for waiting for that channel with the longest MIX-to-MIX delay, but this does not appear in our formulas since we use the maximal delay D_{MtM} anyway). So the only requirement for actual transmission is

(2.2tra) $m \cdot k \leq b$,

where k and b have their usual meaning. For choosing k and b we must now distinguish between networks of different sizes (note that now a stream cipher is used instead of RSA).

Scenario 2A: Worldwide network

Here $D_{tra} = 0.1$, $D_{MtM} = 0.001$ and U/r_{tra} and T are small compared with D_{MtM}. So we take

$k = D_{MtM} = 0.001$

and

$b = a - D_{tra} = 0.1$.

Then we have

(4.2Atra) $m \leq 100$

This requirement is stronger than that derived from connection set up. It means that each message cannot use more than 100 MIXes.

Scenario 2B: Local area network

Here D_{tra} and D_{MtM} are smaller than in scenario 2A. As $b \geq 0.1$ and $k \leq 10^{-4}$, we do not get a better bound for m than (4.2con), so at most 320 MIXes can be used per circuit.

Scenario 2C: Adding dummy traffic

If we want to take a higher percentage of stations as MIXes, we can again gain traffic security by adding end-to-end dummy traffic.

By formula (1) and consequently also from (5'.x) and (5".x) it is easily seen that in order to take more MIXes by a factor of f we also have to increase the rate of events by a factor of f.

Especially, if all stations want to act as MIXes for telephony, and users want to choose arbitrary sequences of MIXes, we must increase the traffic by at least a factor of 2000.

Scenario 3: Telephony with packet switching

Here we use the original MIX scheme with RSA not only for connection set up, as in scenario 2, but for every packet. Again we take $k = 1/32$, but this time combined with $a = 0.2$. For a worldwide network with $D_{tra} = 0.1$ (scenario 3A) this implies $b = 0.1$ and therefore

(4.3A) $m \leq 3.2$

and for any network (scenario 3B) it implies $b \leq 0.2$ and therefore

(4.3B) $m \leq 6.4$.

So for each telephone call at most 6 MIXes can be used, and in a worldwide network only 3, which is much worse than if we use channel switching.

With requirement (1) we will first be very generous, i.e. we will not consider that an attacker can correlate the times when telephone calls start at the senders and receivers (see 2.1.2). Then the events for which traffic security must be guaranteed are just the sending of packets. Their frequency corresponds to a rate

$$r_{ev} = r_{con} \cdot d \cdot \frac{r_{sen}}{U}$$

where r_{con} is the rate of connection set up and d the expected duration of a telephone call, so

$$r_{ev} = 10^{-4} \cdot 180 \cdot \frac{64000}{1000} = 1.152$$

So for a worldwide network we get

$$(5'.3A) \quad M \leq \frac{1}{2} \cdot 0.1 \cdot 1.152 \cdot N = 0.0576 \cdot N$$

and

$$(5''.3A) \quad M \leq \frac{1}{2} \cdot \frac{0.1^2 \cdot 32}{4} \cdot 1.152 \cdot N = 0.04608 \cdot N .$$

(We see that our estimates leading to (5'.x) were not very sharp: Of course the bound derived from (5".x) also holds for the case of arbitrary sequences of MIXes.)
For an arbitrary network we only have to change b from 0.1 to 0.2, so

$$(5'.3B) \quad M \leq 0.1152 \cdot N$$

and

$$(5''.3B) \quad M \leq 0.18432 \cdot N .$$

All these formulas seem to allow a rather large percentage of MIXes among the users (or make not too much dummy traffic necessary if all users want to act as MIXes). But in reality we should not risk the statistical attack on this scheme.
A countermeasure would be to start and finish telephone calls

only at the beginning of fixed time intervals (longer than t, so in contrary to channel switching, this must now be supervised by the user stations). If the delay D_{MIX} is fixed (which is not too difficult to achieve if each MIX adds a time stamp read by the next MIX) and because all packets use the same number of MIXes, all the receivers get the first packet of a call a fixed amount of time (which varies only by the delay between the last MIX and the addressee) after the beginning of each time interval. The same applies to the last packet of a call.
The anonymity guaranteed by this scheme is not easy to compare with that of channel switching (if a scheme with only three to six MIXes per message is to be considered secure at all), especially if different routes are used for different packets of one call:
On the one hand it is an advantage that the beginning and end of a call cannot be recognized at the MIXes, only at the sender and recipient, on the other hand the attacker can take the length of calls into account and, if different routes are used, it is possible that an alternative which was possible for one packet becomes excluded by the next.

2.3 Heterogeneous networks

In the preceding sections we remarked on implementations of the basic concepts for anonymous networks without considering that not all traffic flow in an ISDN is sensitive or must be equally unobservable. Both may be substantiated by examples.

1. If every household gets its electricity bill by the electric power-supply company via the ISDN, only the contents of the bill, but not its existence, gives information to an attacker.
2. In spite of the necessary unobservability of a TV-show's receivers, its sender need not be unobservable.
   More generally, mass communication by its very nature is heterogeneous (or more precisely: asymmetric) in two respects: only its reception must be unobservable and the bulk of traffic only flows in one direction: to the users.

Therefore it seems worthwhile to examine whether we can design

cheaper networks by exploiting these traffic characteristics.

Nevertheless we want to add a warning:
It has been suggested in numerous discussions (e.g. with David Chaum) that, in order to save transmission expense, users should choose for themselves which of their communication relations (also of those which are not common to everybody) are sensitive and which are not and should only use unobservable communication for the first kind. We do not endorse this for several reasons:

- Most data gained by traffic analysis (just like other personal data) are not "sensitive" by themselves, but can be combined with lots of other data to deduce personal profiles or really sensitive data.
- Privacy should not depend on the personal budget (because probably unobservable communication would be more expensive, some people might feel they cannot afford it even if they do consider their communication as sensitive).
- The very fact that someone chooses unobservable communication might arise suspicion.

But let us now examine the possible savings due to special treatment of unsensitive traffic in networks that are designed for anonymity:

If MIXes are employed to make sender and recipient unlinkable, unsensitive traffic need not pass any MIXes, which decreases the number of required MIXes in the network as well as the transmission capacity which must be provided by the physical network. For the latter statement we assumed that the switching centers of the network are not identical with the MIXes (cf. section 2.2.3).

If the RING-network or DC-network are employed to provide sender and recipient anonymity, constant visible implicit addresses may be used for unsensitive traffic to decrease addressing and address recognition expense.

If the DC-network is employed to provide sender and recipient

anonymity, sender anonymity can be dropped for some part of the bandwidth. This saves key distribution or generation and superposition for that part and allows the use of very efficient special purpose medium access protocols (e.g. fixed assignment for TV-stations) or at least the use of nonanonymous general purpose medium access protocols to manage that part, if nonanonymous medium access protocols are more efficient than anonymous ones.

Since for the foreseeable future TV will provide the bulk of traffic users will wish to receive in any ISDN, it is important to provide recipient anonymity as efficiently as possible.
As we noted before, both the DC-network and the RING-network use broadcast to achieve recipient anonymity, which is the only reasonable concept. Therefore, each efficient implementation of the DC- or RING-network efficiently provides recipient anonymity although not necessarily to the highest degree.
In the RING-network, neither the full/empty bit in the case of a slotted ring nor token passing in the case of a token ring is needed for that part of the bandwidth which is dedicated to mass communication, since fixed assignment may be used.
In the DC-network, where some implementations, e.g. star networks, can equip stations with different bandwidth for reception and sending, money may be saved by using sending equipment with less bandwidth. But please note that if we build an asymmetric network on the physical layer we cannot have a movable boundary between these different types of traffic by changing the medium access protocol of the data link layer. The latter may be very useful if we have a lot of interactive business communications during the day and can use the same bandwidth for TV in the evening.

Please note that in about a decade the technical development of coherent detection [BaBr_85, Stan_85] will enable very cost-efficient very high performance implementations providing recipient anonymity, and that even today WDM [Unge_84] allows to overcome limitations of electronic pulse-forming and driving circuits. Therefore network designs which

a) treat transmission bandwidth as a very scarce resource, i.e. the network is only able to transmit that amount of bits to the user station which it really needs and the user station is not able to select the bits it has to process out of a high number of bits without interest to its user, and as a consequence
b) propose networks by which users are completely observable even much more technically obsolete than they are today (cf. section 2.2.1.2).

Network designs which minimize transmission bandwidth and maximize user observability are BIGFON [Brau_83, Kanz_83] and the IBFN planned by the german PTT [Schö_84, ScSc_84], both operated on a "transmission on demand" basis even for the classical broadcast services TV and radio.

2.4 Hierarchical networks

As mentioned in section 2.2.1 and 2.2.2, in the foreseeable future networks which provide sender and recipient anonymity cannot be built for the number of stations an ISDN would have, if each user station is allowed to send 64 kbit/s for telephony, not to speak about 34 Mbit/s for video telephony. The same is true if we restrict our requirements to recipient anonymity alone, as the following calculation shows:

Let each user station receive 100 Gbit/s and 50 million users join the network. As experience suggests, 20 million users might use the network at the same time. If each user sends more than 5 kbit/s, the network will be heavily overloaded.

Since we can hardly drop recipient anonymity (cf. sections 1.1 through 1.4, and 2.3) and have to achieve high performance, it seems reasonable to divide the network stations statically or dynamically into groups which perform broadcasting and one of the schemes of paragraph 1.4 and to support the possible groupings by a physical structure.

This physical structure may have two or more hierarchical

levels. Together with appropriate medium access protocols, it has to support packet and channel switching.
Just like packets are forwarded hop by hop in any hierarchical network, channels are switched by concatenating channels of the different levels of the hierarchy. Therefore, we will concentrate on packet switching and we will only mention channel switching where hierarchical networks enable optimizations or make optimizations more attractive.

2.4.1 Switched/broadcast network

The most reasonable way of defining the quality (as far as unobservability is concerned) of an ISDN that shall provide sender and recipient anonymity and where the special nature of stations is not yet known seems to be by the number of stations amongst which a sender or recipient can be hidden at given cost.
With this definition the best kind of ISDN seems to be a switched/broadcast network structure (SBNS), a two level hierarchical network. At the lower level, stations are statically grouped into physical local area networks, on which broadcasting and one of the concepts for anonymous sending are performed. At the higher level, these broadcast subnetworks are connected by an arbitrary switched network as backbone (figure 8) [Pfit_83, Pfi1_83, Pfit_84, Pfit_85, PfPW_86].
The broadcast subnetworks comprise as many user stations as cost (determined by the available technology) and service expectations of users permit.
Adresses comprise two parts, an outer explicit address and an inner implicit one. Implicit addresses are used to distinguish stations attached to the same broadcast subnetwork, whereas explicit addresses are used in the switched backbone network to select the broadcast subnetwork of the addressee. By this two level hierarchical addressing scheme, broadcast in the backbone network is avoided.

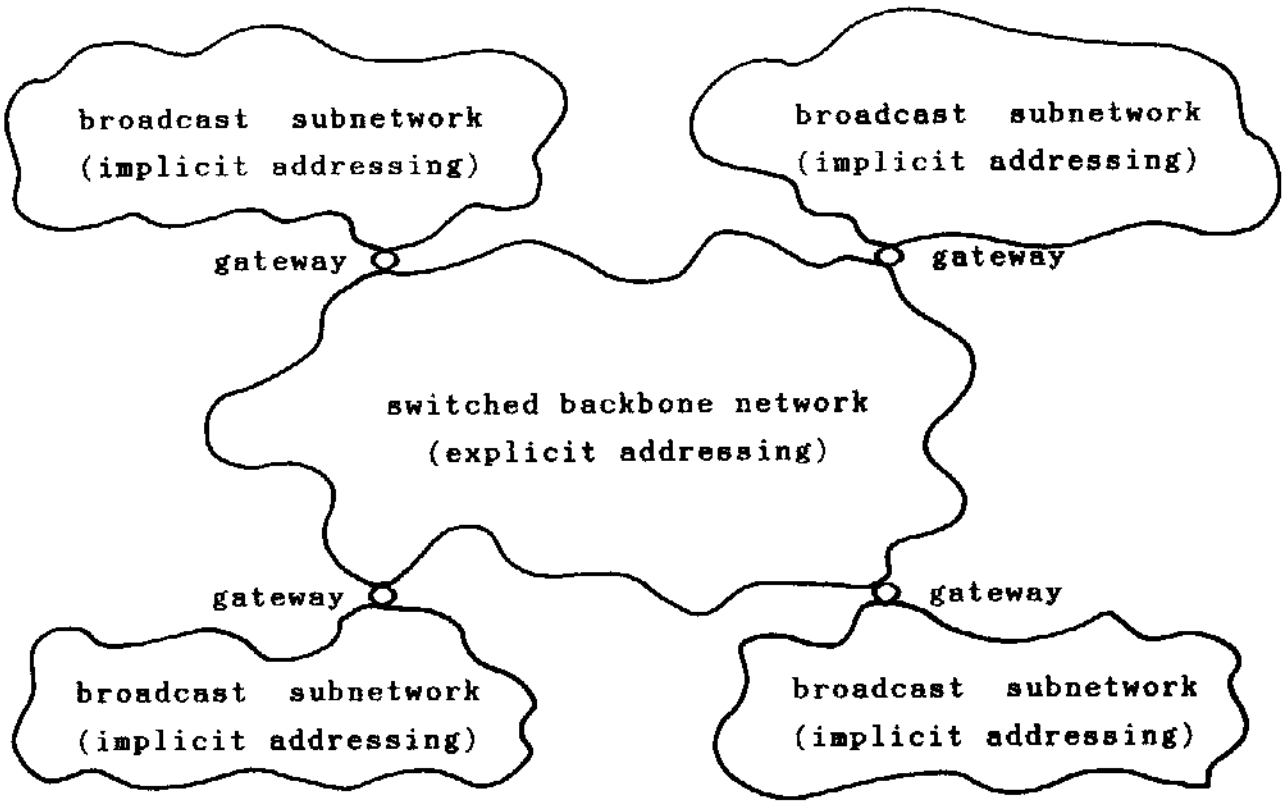


Fig. 8: Switched/broadcast network

The broadcast subnetworks are statically fixed, which offers two advantages: They are easily and efficiently implemented physically and the intersection of the user stations described by different explicit addresses used for the same user station comprises all user stations of that broadcast subnetwork. So there is no possibility of making mistakes in generating addresses, i.e. of giving different kinds of addresses (whose intersection designates exactly one station) to someone during the same correspondence.
Some details of the operation of the subnetworks (RING- or DC-networks) deserve detailed examination. In both cases, there will be one or more gateways to the switched backbone network. These gateways both receive and send a significant fraction of the subnetwork's traffic. Since both actions do not require any unobservability, some optimizations of the basic implementations of the RING- and DC-network can be obtained.

If a RING-network using slotted ring as medium access protocol is employed as broadcast subnetwork, each gateway can empty slots whose content it received correctly and which it forwards to the switched backbone network. If the gateway has something

to send, it may use the emptied slots immediately. A combination of this and the technique of section 2.2.1.1 4) enables very efficient circuit switching throughout the SBNS.

If a DC-network is employed as broadcast subnetwork, the superposition of keys should be done in such a way that a gateway is the first to get the result. In this case, the gateway is able to decide what should be broadcasted to all stations of the subnetwork and what can at once be forwarded to the switched backbone network, thereby optimizing the use of the inbound-broadcastchannel to the user stations. The specific action of the gateway depends on the anonymous medium access protocol which manages the outbound-DC-channel from the user stations.
If slotted ALOHA is used, the gateway has to broadcast whether a successful transmission took place in each slot. If the content of a slot is addressed to one station of the subnetwork, the gateway has to broadcast the content of the slot, of course.
If a reservation scheme is used, the gateway has to broadcast the outcome of bits used to signal reservations. If the reservation scheme does not avoid collisions completely (cf. sections 1.4.1 and 2.2.2.2), the gateway additionally has to broadcast whether a successful transmission took place.
Generalizing these two examples we state that only the control information required by the protocol under consideration must be broadcasted by the gateway. The bandwidth saved by suppression of information intended for other subnetworks or of garbled information may be used to broadcast information arriving from other subnetworks as well as mass communication.
Of course, this scheme is only effective, if the control information is only a minor part of the transmitted information as in the above mentioned protocols as well as in R-ALOHA, ARRA and the collision-resolution algorithms (all mentioned in section 2.2.2.2). Therefore, this scheme is inefficient for CSMA protocols (cf. section 2.2.2.2).

If appropriate physical ring networks are used as broadcast subnetworks, their bandwidth can be shared between a RING-network

and a DC-network (so we get a heterogeneous network, cf. section 2.3). In the local area, this allows a combination of the efficiency of the former with the powerful unobservability of the latter.
Additionally, for services with low performance requirements superposed sending can be performed in larger groups than those supported by the physical structure of the network.
Both measures may be performed in a way that the users classify the traffic originated by them into a more sensitive and a less sensitive class and choose the appropriate part of the bandwidth accordingly, but the warning in section 2.3 applies (though in a weaker form). Alternatively the classification can be done according to the service used, e.g. all broadband communication uses the RING-network part, all telephony uses the DC-network part in the local broadcast subnetworks and all electronic mail uses superposed sending in groups comprising ten local broadcast subnetworks.
Another extension may be that MIXes foil traffic analysis of some classes of data in the switched backbone network. Of course, as far as enough noncolluding MIXes, enough crypto capability at user stations, and enough bandwidth are available, all traffic passing the switched network may be mixed.

As noted in section 1.5, the MIX-, DC- and even the RING-network can be considered to be virtual concepts. The same is true for the SBNS (and all networks which will be described in the following).
Therefore one may ask why we group the stations of users which are only close apart from each other in a broadcast subnetwork implemented by a local area network instead of grouping the stations of users which are similar to each other in some respects. For example, senior students work some hours later than most other people or read other newspapers than blue collar workers. If just one tabloid is ordered at 6 o'clock in the morning in a subnetwork which connects 999 students and one blue collar worker, it seems to be quite clear who ordered the tabloid.
The answer to the question above is that local broadcast

subnetworks can be implemented much more efficiently than wide area broadcast subnetworks, so we would hope that we could push technology not only to include 999 students but 999 blue collar workers as well in an advanced version of our local broadcast subnetwork mentioned in the example. This seems more promising than grouping similar users in small "virtual" broadcast subnetworks. Especially, we can neither define what are similar users (users which are similar in some respect may be quite dissimilar in another) nor have we an accepted quantitative definition of unobservability taking similarities of users into account.

2.4.2 Broadcast/broadcast network

If we think that an attacker is not able to control too many gateways, we may want to foil traffic analysis at the second level of the hierarchy as well. Then the concept of a broadcast/broadcast network as depicted in figure 9 springs to mind, since under the above assumption it can provide nearly the unobservability of a nonhierarchical (flat) DC- or RING-network at significantly less cost [Pfi1_83 pp. 66, 67]. The reason for the last statement is that most lines in the network (all lines in the subnetworks) and nearly all stations (all user stations) do not have to operate with the speed of the broadcast backbone network and gateways.

For the foreseeable future, though, the performance of a backbone broadcast network which can be provided at reasonable cost cannot accommodate the demand of a national ISDN, not to speak about an international one.
Therefore, broadcast/broadcast networks can only be used on some channels for services with low performance requirements (cf. section 2.4.1) or as subnetworks e.g. in a switched/broadcast/broadcast network or, more generally, in a switched/(switched/)*(broadcast/)*broadcast network (using Kleene's star notation [WaGo_84 p. 117]).

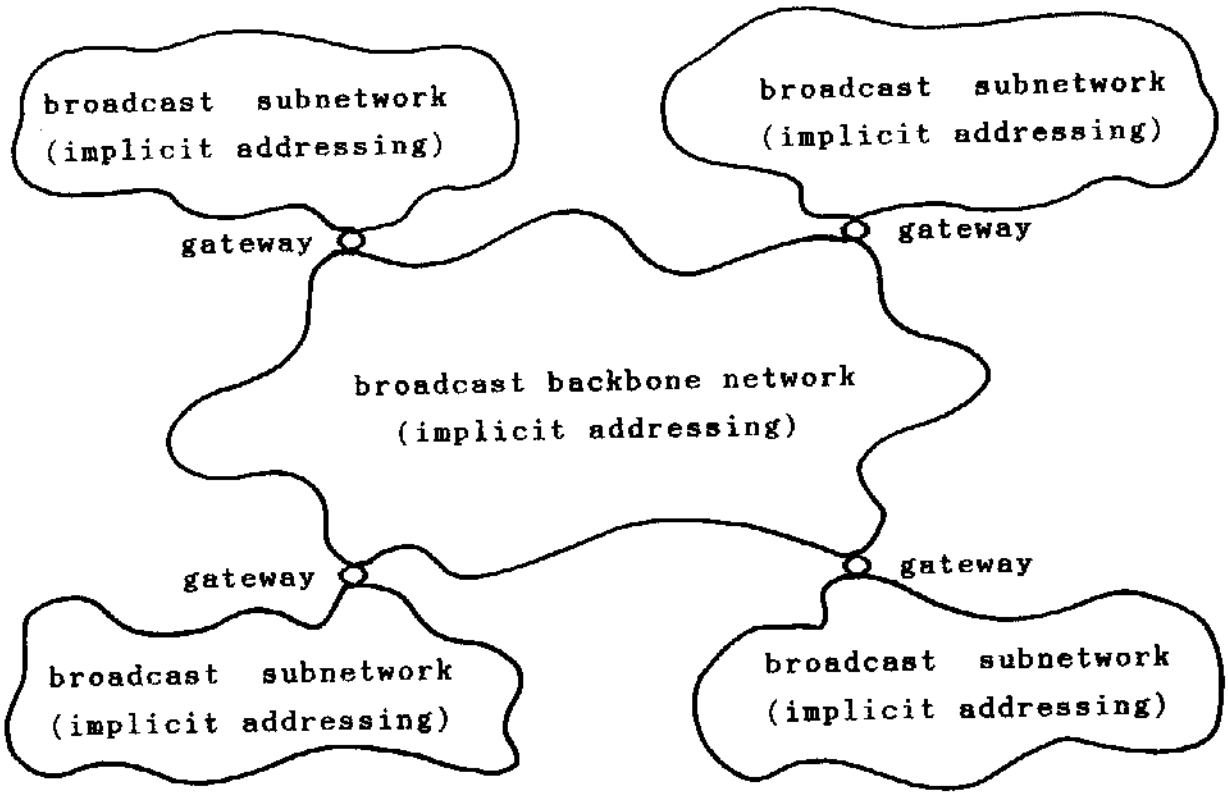


Fig. 9: Broadcast/broadcast network

In the case a RING-network is used as implementation of the backbone broadcast network, an attacker must not only be prevented from controlling too many gateways but also from controlling too many lines. Otherwise he would get as much information as in the cheaper SBNS. As it seems difficult to protect long-distance lines physically, virtual-link-by-virtual-link encryption must be used. But the fact that no link-by-link encryption is needed was just one of the advantages of a RING-network.

If a DC-network is used as implementation of the backbone broadcast network, tapping of the lines gives no information at all to an attacker, so no additional protection for the long-distance lines is necessary.

Still in both cases, if the gateways are all operated by the same operator and have the same manufacturer, as will probably be the case in a state with a telecommunication monopoly as in the FRG, the manufacturer must be controlled not to install Trojan Horses and the operator must be prevented from causing any harm by placing the gateways into tamper-resistant modules

(cf. section 1.1). Both measures may be rather expensive.
If these conditions can be satisfied ("satisfied" here has a meaning of social acceptability and adequacy of trust to risk; if these conditions would be satisfied in an absolute sense, we would only need link-by-link encryption to reach unobservability of users in nearly any network), the broadcast/broadcast network becomes a cost-efficient [Sze_85] alternative to "flat" broadcast networks as subnetwork in switched networks.
If these conditions cannot be satisfied (or the users are not convinced that they are satisfied), we have to concentrate on building as sizable and efficient "flat" broadcast subnetworks as possible or build broadcast networks whose size can be chosen dynamically by the users as explained in the next two sections.

2.4.3 Tree network with dynamic key graphs

To be able to partition the broadcast network dynamically, we can physically build a tree network [Yemi_83, SuSY_84, LeBo_83, BoLe_83, Alba_83, ClLe_81].
We slightly modify the routing rule to partition the tree network concerning the reception of messages:

Each message M travels first in direction of the root until it reaches a node F, which is known to be a father of the addressee.

F stops M traveling to the root and broadcasts it to all its sons.

The sons broadcast M to their sons and so on.

Figure 10 shows a possible situation, where four intermediate nodes each broadcast a particular message to their sons.
That way, we increase the throughput of our broadcast network concerning the reception by a factor of 4 in the situation of our example, but decrease the number of stations amongst which a recipient can be hidden accordingly.

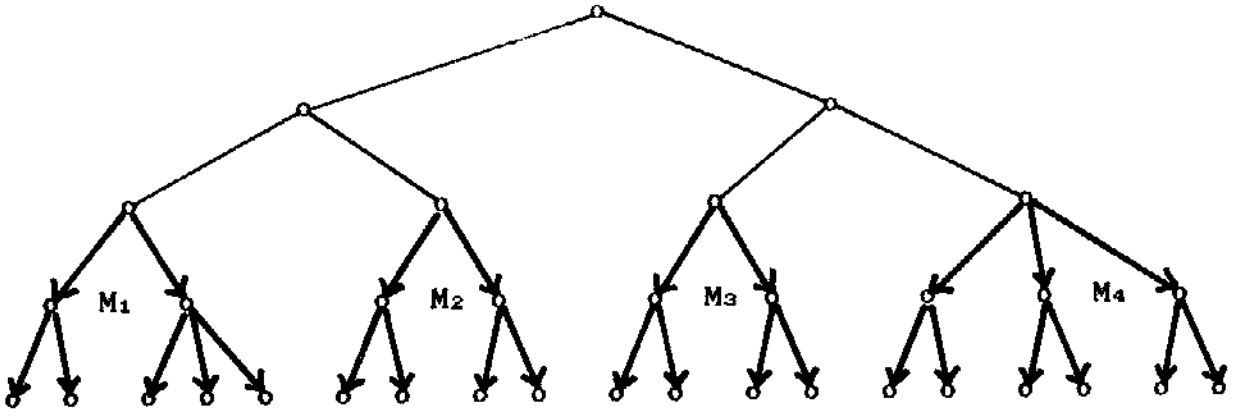


Fig. 10: Tree network with dynamically partitioned broadcast

To achieve anonymity of the sender, superposed sending is used. The additions mod 2 are done on the way towards the root.
To achieve higher throughput than in a plain broadcast network not only concerning the reception of messages, but concerning the sending of messages as well, we can use a time division partitioning of the tree and appropriately chosen dynamic key graphs.
In the first time partition (potentially) global (e.g. international) traffic takes place: all messages travel to the root and are broadcasted world-wide. Keys for this time partition can (and should be) shared with other user stations all over the world.
In the $n+1^{st}$ time partition, all messages travel only to the n^{th} sons of the root (representing e.g. continentals, states, districts, ...). Keys for these time partitions are only shared between user stations which are sons of the same n^{th} son of the root.
After the most local traffic is served, the most global traffic is served again.

All medium access protocols described so far can be used to manage the bandwidth of each partition, albeit with different efficiency. CSMA, for example, seems not to be suited to international traffic.

If it seems appropriate, the sequence and length of partitions can be dynamically adapted to the varying demands of users, characterized by their present traffic volume, its possible locality (most local time partition where sender and addressee are in the same partition) and the locality chosen by the user station for it (possibly a more global time partition).
If the more global time partitions or the tree network as a whole is only slightly used, user stations can transmit messages in a more global time partition than necessary to use the superfluous bandwidth to further decrease user observability. Because the intersections of subtrees are empty or the smallest one, there are no possibilities to make mistakes in chosing time partitions, which corresponds to chosing explicit addresses.

Of course, the global time partitions will be overcrowded most of the time, since our tree network operates at its root with exactly the same bandwidth as do all user stations, and all international traffic has to pass through the root.
Therefore, it is necessary to get rid of this bottleneck by combining our tree network with the concept of the SBNS, that is switching in the higher level(s) of a network. This will be explained in the next section.

2.4.4 Switched/tree network

If the scheme of superposed sending is used, the SBNS can easily be combined with the tree network with dynamic key graphs yielding a switched/tree network. The partitioning into local broadcast networks can then be made variable by changing the depth of the switched backbone network as depicted in figure 11.
The depth of the switched backbone network may depend on the volume of traffic and/or the sensitivity class of the traffic. As explained in the preceding section, different time partitions may be used to accommodate different sensitivity classes quasisimultaneously.
The price to be paid for this adaptability is, that all nodes potentially at the border between broadcast tree network(s) and

switched backbone network must be able to act as gateway. That is the reason why we depicted these nodes by larger circles in figure 11.

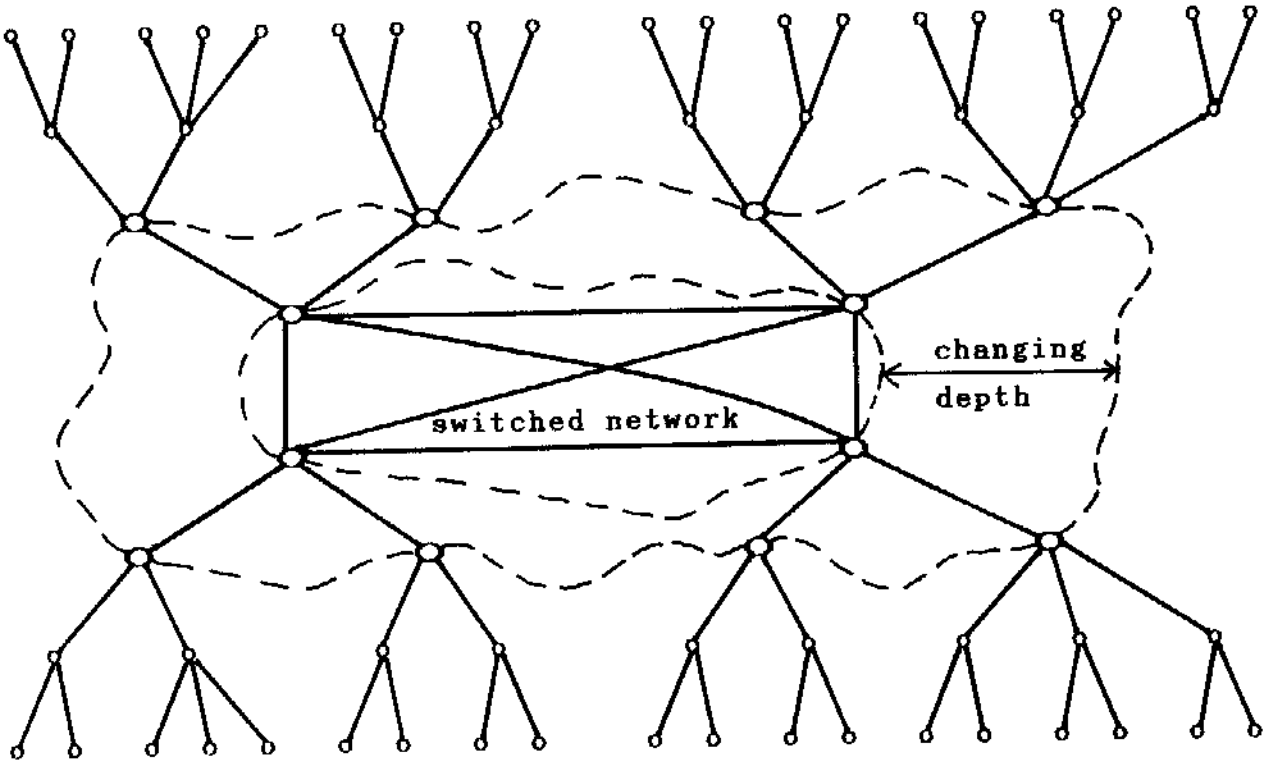


Fig. 11: Switched/tree network

The wish of some users to be protected by other stations than their physical neighbors can be supported by the network in case of superposed sending on tree structures if the backbone network comprises a tree structure, too, because then the border between local broadcast networks and the global switched network can be chosen arbitrarily near the center of the network. So the network can be operated in several modes, where of course the most anonymous one offers the smallest bandwidth to each user.

3 Fault tolerance

So far, all networks without user observability are serial systems in the sense of reliability: all MIXes of a chosen sequence of MIXes, all stations of a RING, and all stations taking part in superposed sending have to work correctly. To fulfil the high reliability requirements on an ISDN, each scheme must be extended to include some fault tolerance mechanisms. These mechanisms may work end-to-end, i.e. the sender retransmits a message if it does not receive an acknowledgement after a certain period of time. Even if the sender chooses a different encoding of the message for each retransmission, the retransmitted messages may enable statistical attacks in some networks. Moreover, the performance of such mechanisms in terms of average transfer delay, variance of transfer delay, or usable throughput may be unsatisfactory. Therefore, it seems worthwhile to use mechanisms which avoid end-to-end retransmission wherever possible.
Moreover, retransmission only helps if the fault is transient (or if the retransmission can circumvent it). To cope with permanent faults, reconfiguration and error recovery are necessary.

Before considering the MIX-, DC- and RING-network, we want to remind of section 1.5 on layering.
As mentioned there, the medium, physical layer, data link layer and the lower part of the network layer of the MIX-network as well as the medium and the lower part of the physical layer of the DC-network may be implemented arbitrarily. Since we do not intend to write a survey on the implementation of fault tolerant networks in general, we will only note that in these (sub)layers, all fault tolerance mechanisms deemed appropriate may be applied without any restrictions caused by anonymity requirements and that layered systems can easily be constructed in such a way that faults (which cannot successfully be handled in the layer in which they occur) only propagate to higher layers (which may handle them) and never to lower layers.
The upper sublayer of the network layer generates anonymity in

the MIX-network as do the upper sublayer of the physical layer in the DC-network and the medium together with digital signal (re)generation in the RING-network. Since unreliability of these (sub)layers causes end-to-end retransmissions if we do not add fault tolerance mechanisms to these (sub)layers or the layers close above, and in these layers inappropriate fault tolerance mechanisms could destroy anonymity, we have to design or adapt appropriate mechanisms.

Since the application, presentation, session and transport layer entities only concern the service for a single user and usually reside in the user equipment (which can be assumed to function properly, since otherwise the user cannot expect any service) and their actions are at least partially time-decoupled from the much faster events at the lower layers and all user stations seem to be equal to each other (cf. the summary of anonymous medium access protocols in section 2.2.2.2) we suppose that these layers will not pose difficult reliability or anonymity problems. Due to restrictions in our time and paper budget, we will therefore concentrate on the remaining layers.

3.1 MIX-network

In the MIX-network, fault tolerance is especially necessary and useful in the case of return addresses where the end-to-end timeout and retransmit approach does not work satisfactorily for all services, especially those where no reasonable timeout is possible (e.g. electronic mail). The reason for this is that only the original sender, not the sender of the return message, can construct another address as replacement for an unusable one, if a MIX breaks down which must be passed. If end-to-end retransmission has to be used in this case, every user station has to monitor the availability of every MIX which will be passed by a message guided by an outstanding untraceable return address and send another untraceable return address if it becomes unavailable.

To provide fault tolerance in the sublayer implemented by the

MIXes, there are three obvious possibilities:

1) Let every station generating an address incorporate multiple diverse sequences of MIXes and let every station using an address try one sequence and after a timeout the next one (dynamic redundancy) or all sequences in parallel (static redundancy).
2) Let every MIX choose one or more MIXes as backups for itself, which receive its private key and are announced to all other MIXes to be its backups.
   If a certain amount of time can be spent on establishing the backup service, the backup MIXes may receive only parts [Sham_79] of its private key. If the MIX under consideration fails, these MIXes must cooperate to establish the backup facility. This guarantees that, as long as no quorum of the other MIXes cooperates, no other MIXes can compromise the unobservability provided by the MIX under consideration.
3) Let every MIX get enough information by each message to skip one MIX (or in general: at most a fixed number of MIXes).

In the following, we will examine and evaluate these possibilities for the case that messages (or packets) are transmitted.
If a channel is to be switched, the same techniques may be used during channel set up. If the channel is in use, any failure of a MIX will cause at least a short interruption of service noticed by the receiver. Since at least one user of the channel will notice its interruption anyway and the failure of a MIX during channel usage is infrequent (given that channels are not used permanently), the interrupted channel may be abandoned and a new one established. For this, even receivers of simplex channels must get return addresses, which can either be used for regular acknowledgements or for signalling in case of detected faults (in the latter case, there should be fault tolerance for the return addresses, too).
This end-to-end mechanism seems to be quite efficient for infrequent interruptions and poses no special problems concerning unobservability, if the release of the damaged channel (more precisely: the two parts left of it) follows the rules described for the normal release of channels in section 2.1.2. Of course,

the new channel can be established before the old one is released.
If the duration of an interruption must be kept very short, the possibilities 2) or 3) may be used to shorten the interruption, but then additional problems concerning unobservability are encountered which are similar to those explained in the context of packet switching for the possibilities 2) and 3) later.

3.1.1 Diverse sequences of MIXes

The first possibility decreases the unobservability only a little bit: the scheme is broken iff one of the multiple diverse sequences can be traced by an attacker. So this possibility gives an attacker multiple different tries, but each try is as difficult as the original scheme.
But the first possibility suffers from two disadvantages: it can only work end-to-end and there must be tremendous numbers of diverse sequences if the length of each sequence is great and therefore its reliability is not too close to 1. This phenomenon concerning the reliability of parallel-serial systems is derived mathematically in [PfHä_82, Pfit_82].

3.1.2 MIX replacement schemes

In the second and third possibilities, we equip some MIXes with equal capabilities, so they can replace each other. This is a great gain concerning reliability (esp. these possibilities have neither of the disadvantages of the first possibility) but poses a big problem concerning unobservability (called coordination problem), which is treated in the next subsection.

3.1.2.1 The coordination problem

In the original MIX scheme, each MIX was responsible never to mix the same message twice (we only consider a time interval in which the MIX does not change its key pair, cf. section 1.3). Otherwise, an attacker could bridge the MIX with respect to the message it mixed twice, since with very high probability it will be the only message which appears twice in the output of the MIX. Our fault tolerance mechanisms distribute this responsibility to a team of MIXes, which must satisfy this responsibility even if some members of the team are down. Even messages mixed by the members of the team which are down at the moment must not be mixed by the other members of the team a second time in spite of an attacker who will strive to cause this.
There is a simple protocol guaranteeing that every message is mixed at most once by the members of the team. But the overhead (in termes of additional delay, communication between and memory inside MIXes) caused by this protocol is severe.

Inefficient coordination protocol:

Before mixing a batch, a MIX submits the input messages to all other members of the team which are currently up. The MIX mixes a message only after it received confirmations by all members of the team which are currently up that they have never mixed it before. A MIX which was down gets from the others all messages which were ever mixed or are intended to be mixed, before it participates in mixing again.

Of course, each MIX of the team must store all messages mixed by the others as well as those mixed by itself. To get the memory requirements of this protocol reasonably small, the techniques mentioned in [Chau_81] for the case of single MIXes may be used: Keys may be changed often or messages may include in their random string a date and time interval in which they may be mixed. After expiration of this time interval, they may be forgotten by all MIXes of the team.
Another mechanism to reduce the memory requirements is to apply a compressing one-way function [DaPr_84] to each message and to store and compare only images of messages under that one-way function. This reduces messages of some thousand bits or even

more to images of about 100 bits. The price to be paid is that with a very low probability two different messages will yield the same image and the MIXes will not mix the second one, which the original sender must encode using other random bit strings and another (return) address and retransmit end-to-end.
If the one-way function is applied by the sending (and not by the receiving) MIX, this mechanism not only reduces the memory requirement but the communication overhead as well.
Other mechanisms to reduce the communication overhead and delay incurred are specific to the second or the third possibility, respectively, and will therefore be considered later.
The protocol described above presupposes that MIXes know reliably which MIXes are down, since otherwise an attacker may try to isolate two groups, telling both that all MIXes of the other group are down, and then mount his attack. To foil this, the MIXes of a team should only mix if a majority is up and cooperating. This may be guaranteed by simple authentication techniques.

3.1.2.2 MIXes with backups

The second possibility needs no modification of the address nor of the message format. The MIXes have to manage the redundancy, that is the private keys (or parts thereof [Sham_79]) and routing adaptive to the network configuration formed by the MIXes which are up (and the state of the underlying communication system, which we will ignore as explained at the beginning of section 3). To manage this redundancy, each MIX verifies that the next MIX is up and processed and transmitted the packet successfully by a MIX-to-MIX protocol.
As announced, the overhead of the protocol which prevents multiple mixing of the same message can be drastically reduced.

Efficient coordination protocol for MIXes with backups:

Any delay may be avoided, if each MIX agrees with its backup MIXes that they only mix with its key if it is down. Then it suffices that it sends the corresponding input messages of all mixed messages (of all batches) to a sound majority of its

backup MIXes at the same time when it outputs the batch. If it stops mixing if it does not get a confirmation of receipt by a majority of its backup MIXes, an attacker can at most bridge that MIX concerning the messages for which it did not get receipts yet by isolating it from the other MIXes. Backup MIXes which the MIX under consideration is able to verify as down (e.g. it can communicate with and authenticate a front end processor of that MIX) need not be considered in determining a majority.

If a MIX is down, its backup MIXes may designate one of them as its agent (and enable it to reconstruct the key, if a threshold scheme like [Sham_79] is used). The agent acts exactly like the MIX would do if it were up, until it either fails as well (and the backup MIXes designate a new agent) or the MIX that failed is repaired and resumes mixing (synchronized with the agent's stopping).

We think that the risk that an attacker can bridge a MIX concerning those messages for which that MIX did not get receipts yet by isolating it from the other MIXes is far outweighted by the avoidance of any additional delay, because active attacks trying to exploit this flaw are detected easily (at least if their frequency is high) and must be very sophisticated to reveal any interesting information if several MIXes are used per message: To trace the way of a particular message from its sender to its recipient an attacker has to (control or) disable or isolate one MIX of every backup team (of which the message must pass one member) directly after it mixed that message.

A MIX should choose MIXes as backup which are physically separated at least a few tens of kilometers, to foil as many common mode failures as possible, e.g. floods, electricity outages, rockets flying amuck, and the like.

If one organization is operating several MIXes at places which are far apart, it seems reasonable that these MIXes mutually backup each other.

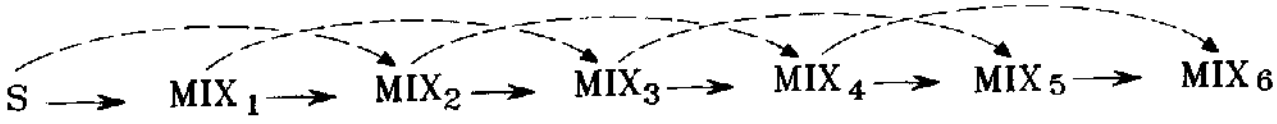
3.1.2.3 Skip MIXes schemes

In contrast to the second possibility, the third needs modification of address and message format. These will be described in 3.1.2.3.1. In 3.1.2.3.2 we state the security that can be reached if these schemes are properly used.
The redundancy that all these schemes provide may be operated in one of two modes: in the mode "skip as few as possible" only those MIXes are skipped which are down, whereas in the mode "skip as many as possible" in each step all MIXes which can be skipped are skipped. That means that each MIX passes the message to the farthest of those MIXes which are up and which it can reach due to the fault tolerance scheme.
These modes and how they influence the coordination problem and thus the security are described in turn in sections 3.1.2.3.3 and 3.1.2.3.4.

3.1.2.3.1 Message and address formats

To ease understanding, we will first describe the case that every MIX can bypass the next MIX in a sequence of chosen MIXes. In this case, a failure of one MIX (or more, as long as no two consecutive MIXes break down) is tolerable.
To bypass one MIX, its predecessor must not only get the message part for it but also for its successor (figure 12).
If it receives both message parts and this is done for every MIX, the length of the whole message grows exponentially with the number of MIXes. To avoid this, the sender of a message chooses a different key (e.g. of a less expensive symmetric cryptosystem) for each MIX, with which this MIX decrypts the message. Together with its message part, each MIX has to get its key and that of its successor, both together encrypted with its own public key. The addresses of the next two MIXes may be prefixed to each message respectively or both addresses may be encrypted together with the two beforementioned keys. In the following, this is described more formally.

general scheme:



——→ first try ----→ second try

example: MIX 2 and MIX 4 are down

Fig. 12: Skip 1 MIX scheme operated in the mode "skip as few as possible"

Let again $A_1,\ldots,A_n$ be the sequence of addresses and $e_1,\ldots,e_n$ the sequence of public keys of the chosen MIXes $MIX_1,\ldots,MIX_n$; A_{n+1} the address of the addressee (called MIX_{n+1} for convenience), and e_{n+1} his public key; $k_1,\ldots,k_n$ the chosen sequence of keys, and M_i the message that MIX_i shall receive. The sender forms the messages M_i according to the following recursive scheme, starting from the message content M that MIX_{n+1} shall receive:

$$M_{n+1} = e_{n+1}(M) \qquad \text{(scheme 1)}$$
$$M_n = e_n(k_n), k_n(A_{n+1}, M_{n+1})$$
$$M_i = e_i(k_i, A_{i+1}, k_{i+1}), k_i(A_{i+1}, M_{i+1}) \quad \text{for } i=1,\ldots,n-1$$

Of course, the described scheme is not the only appropriate one. The scheme given below will do as well.

$$M_{n+1} = e_{n+1}(M) \qquad \text{(scheme 2)}$$
$$M_n = e_n(k_n, A_{n+1}), k_n(M_{n+1})$$
$$M_i = e_i(k_i, A_{i+1}, k_{i+1}, A_{i+2}), k_i(M_{i+1}) \quad \text{for } i=1,\ldots,n-1$$

The sender sends M_i to MIX_i.
In both schemes MIX_i can compute M_{i+1}, M_{i+2}, A_{i+1} and A_{i+2} out of M_i.

With some coordination protocols (example later) it is also important that MIX_i can check that its predecessor MIX_{i-1} did not tamper with the message (and thus also nobody else, because MIX_{i-1} knows most about the message).
As long as the cryptosystems are not broken, an attacker could of course only tamper with message parts encrypted with keys it knows. Otherwise the message just gets lost or is discarded by somebody who finds that it decrypts to garbage (we assume that enough redundancy is added everywhere so that this can be noticed) so the attacker gets no information about the message. So MIX_{i-1} has the possibility to exchange the complete part encrypted with e_i (without being able to read the original one, because he knows e_i, but not d_i) or to tamper with the part encrypted with k_i only.
The danger caused by this is that he might try to change the addresses. This causes no harm to unobservability as long as addresses are only used to forward messages, because the attacker is assumed to control all lines anyway, but may cause harm if the addresses are also used to determine who must coordinate with whom not to mix a message twice.
In both schemes the attacker would have to exchange the part encrypted with e_i to change an address. To do so, he must invent a new k_{i+1}. Then the part encrypted by k_{i+1}, i.e. $k_{i+1}(A_{i+2},M_{i+2})$ in scheme 1 or $k_{i+1}(M_{i+2})$ in scheme 2, decrypts to garbage. So there must be enough redundancy in these parts that already MIX_i can notice this.
If a coordination protocol is used where this tamper-check is unnecessary, the first A_{i+1} in scheme 1 can be left out.

The choice between scheme 1 and scheme 2, i.e. whether we include A_{i+2} in the first part encrypted using an expensive asymmetric cryptosystem or in the second part (under the name of A_{i+1} for i := i+1) encrypted using a less expensive symmetric cryptosystem depends on the block size of the former. If for

reasons of security of the asymmetric cryptosystem, the block size must be chosen great enough to include A_{i+2}, scheme 2 should be used instead of encrypting additional garbage in scheme 1, since encrypting A_{i+2} for free using the expensive asymmetric cryptosystem is cheaper than encrypting A_{i+2} (under the name of A_{i+1} for i := i+1) using the less expensive symmetric one. If A_{i+2} includes any redundancy we must (in principle) be cautious that it does not reduces the entropy of $k_i, A_{i+1}, k_{i+1}, A_{i+2}$ below an acceptable level. But since the symmetric cryptosystem must withstand an exhaustive key search attack, both k_i and k_{i+1} each include enough entropy.

If the block size of the asymmetric cryptosystem can be chosen so small that even the inclusion of A_{i+1} causes the encryption of an additional block (because the block cannot include A_{i+1} or its entropy is reduced below an acceptable level, which is only possible if it does not include k_i and k_{i+1}, i.e. it needs several blocks to include both) we may use a one-way function f [DaPr_84] (and apply it not only to A_{i+1} but to a concatenation with something including a lot of entropy so that the result of the one-way function may include more entropy than A_{i+1}) to hinder MIX_{i-1} from changing A_{i+1} in M_i, resulting in the following scheme:

$$
\begin{aligned}
M_{n+1} &= e_{n+1}(M) \qquad \text{(scheme 3)}\\
M_n &= e_n(k_n), k_n(A_{n+1}, M_{n+1})\\
M_i &= e_i(k_i, k_{i+1}), k_i(f(k_{i+1}, A_{i+1}), A_{i+1}, M_{i+1}) \quad \text{for } i=1,\ldots,n-1
\end{aligned}
$$

Each MIX_i checks whether f applied to what it considers to be k_{i+1}, A_{i+1} results in the first part of the deciphered message it gets.
If MIX_{i-1} wants to tamper with A_{i+1} secretly, it has to find the corresponding $f(k_{i+1}, X)$ for some $X \neq A_{i+1}$. But this is infeasible without the knowledge of k_{i+1}.

Next we have to show how the untraceable return address described in section 1.3 can be made fault tolerant as well.
To enable the recipient to reply in the case of a failure of one

MIX (or more, as long as no two consecutive MIXes break down), the sender may form a fault tolerant untraceable return address (B_1, A_1, k_0, k_1), and include it in a message to the recipient. Here k_0 is a key (of any cryptosystem) chosen for the occasion, by which the recipient should encrypt the return message. A_1 is the address of MIX_1 and k_1 the key chosen for it (so that the recipient can skip MIX_1). B_1 is the actual return address, which (in analogy to usual return addresses) is constructed like a message without message content to the original sender.
So if we use scheme 1, we obtain the following recursion scheme:

$$B_{m+1} = \emptyset \qquad \text{(scheme 4a)}$$
$$B_m = e_m(k_m), k_m(A_{m+1}, B_{m+1})$$
$$B_j = e_j(k_j, A_{j+1}, k_{j+1}), k_j(A_{j+1}, B_{j+1}) \quad \text{for } j=1,\ldots,m-1$$

The recipient uses (B_1, A_1, k_0, k_1) and its reply message content C to form $M_1 = B_1, k_0(C)$, which it sends to MIX_1 if MIX_1 is up. Otherwise, it uses k_1 to find A_2 and B_2 and to form $M_2 = B_2, k_1(k_0(C))$, which it sends to MIX_2.

When MIX_j gets a message $M_j = (B_j, C_j)$, where C_j is the message content part (and B_j the return address part as shown above), it uses its private key d_j to get k_j, A_{j+1}, and k_{j+1} out of B_j.
Now it can form $C_{j+1} = k_j(C_j)$, decipher $k_j(A_{j+1}, B_{j+1})$, and send $M_{j+1} = (B_{j+1}, C_{j+1})$ to MIX_{j+1} if this MIX is up.
Otherwise MIX_j also forms $C_{j+2} = k_{j+1}(C_{j+1})$, deciphers $k_{j+1}(A_{j+2}, B_{j+2})$, and sends $M_{j+2} = (B_{j+2}, C_{j+2})$ to MIX_{j+2}.
During all these steps MIX_j can check that the return address has not been tampered with, like in scheme 1.
To sum up, the message M_j that MIX_j shall receive is derived from the return message content C by the recursion

$$M_1 = B_1, C_1; \quad C_1 = k_0(C) \qquad \text{(scheme 4b)}$$
$$M_j = B_j, C_j; \quad C_j = k_{j-1}(C_{j-1}) \quad \text{for } j=2,..,m+1$$

Only the sender can decrypt $C_{m+1} = k_m(\ldots k_1(k_0(C))\ldots)$, because he created k_0 through k_m.

Our fault tolerant untraceable return address scheme is formed corresponding to scheme 1. Variations corresponding to scheme 2 or scheme 3 may be formed as well.

Next, we want to generalize all schemes from 1 MIX, which, when down, may be bypassed, to k consecutive MIXes which, when down, may be bypassed. This generalization enables us to improve reliability. The fault tolerant message schemes which allow to bypass k successive MIXes may have the following forms. In all three schemes MIX_i for $i \leq n-k$ can compute M_{i+1} through M_{i+k+1} and A_{i+1} through A_{i+k+1} out of M_i, and check that its predecessors did not tamper with the message, because they do not know k_{i+k+1}. If $i>n-k$, MIX_i can compute M_{i+1} through M_{n+1} and A_{i+1} through A_{n+1} out of M_i, and check that those of its predecessors which cannot bridge it anyway did not tamper with the message, because they do not know k_n.

Extension of scheme 1:

$$M_{n+1} = e_{n+1}(M) \qquad \text{(scheme 5)}$$

$$M_i = e_i(k_i, A_{i+1}, k_{i+1}, \ldots, A_n, k_n), k_i(A_{i+1}, M_{i+1}) \quad \text{for } i=n-k+1,\ldots,n$$

$$M_i = e_i(k_i, A_{i+1}, k_{i+1}, \ldots, A_{i+k}, k_{i+k}), k_i(A_{i+1}, M_{i+1}) \quad \text{for } i=1,\ldots,n-k$$

Extension of scheme 2:

$$M_{n+1} = e_{n+1}(M) \qquad \text{(scheme 6)}$$

$$M_i = e_i(k_i, A_{i+1}, k_{i+1}, \ldots, k_n, A_{n+1}), k_i(M_{i+1}) \quad \text{for } i=n-k+1,\ldots,n$$

$$M_i = e_i(k_i, A_{i+1}, k_{i+1}, \ldots, A_{i+k}, k_{i+k}, A_{i+1+k}), k_i(M_{i+1}) \quad \text{for } i=1,\ldots,n-k$$

Since the one-way function may be compressing (100 bits, say, are sufficient as result, whereas the input may be arbitrarily large), the following scheme, which is an extension of scheme 3, is particularly appropriate if k is large.

$M_{n+1} = e_{n+1}(M)$ (scheme 7)

$M_i = e_i(k_i,\ldots,k_n),k_i(f(k_n,A_{i+1},\ldots,A_n),A_{i+1},M_{i+1})$ for $i=n-k+1,\ldots,n$

$M_i = e_i(k_i,\ldots,k_{i+k}),k_i(f(k_{i+k},A_{i+1},\ldots,A_{i+k}),A_{i+1},M_{i+1})$ for $i=1,\ldots,n-k$

Next we have to show how the fault tolerant untraceable return address described (scheme 4) can be generalized to a k-consecutive fault tolerant untraceable return address $(B_1,A_1,k_0,k_1,k_2,\ldots,k_k)$.
Again this is only the scheme derived from scheme 1, thus with the return address looking like a message in scheme 5. Schemes derived from scheme 2 or 3, thus 6 or 7, may be formed as well. The B_j and C_j have the same meaning as in scheme 4.

$B_{m+1} = \emptyset$ (scheme 8a)

$B_j = e_j(k_j,A_{j+1},k_{j+1},\ldots,A_m,k_m),k_j(A_{j+1},B_{j+1})$ for $j=m-k+1,\ldots,m$

$B_j = e_j(k_j,A_{j+1},k_{j+1},\ldots,A_{j+k},k_{j+k}),k_j(A_{j+1},B_{j+1})$ for $j=1,\ldots,m-k$

$M_1 = B_1,C_1;\quad C_1 = k_0(C)$ (scheme 8b = scheme 4b)

$M_j = B_j,C_j;\quad C_j = k_{j-1}(C_{j-1})$ for $j=2,..,m+1$

3.1.2.3.2 Security criteria

For the case that MIXes are properly coordinated never to apply the same transformation to the same message twice, e.g. by the inefficient coordination protocol of 3.1.2.1, we claim that the security of our skip MIXes schemes is as follows. Again we begin with the schemes where each MIX can skip only one next MIX for simplicity.

Security criterion:
If the MIXes are properly coordinated the schemes 1, 2, and 3 are as secure as the original [Chau_81] one as long as either

a) the first MIX which mixes the message or
b) at least two consecutive MIXes of those addressed in the message which (in the case they are both up) can communicate or
c) at least two consecutive MIXes of those addressed in the message and a majority of all existent MIXes

are not controlled by the attacker.

For the fault tolerant return address scheme only condition a) of the security criterion must be changed to "the sender of the reply message and the first MIX which mixes the message or".

Also for the more general schemes where each MIX can skip the next k MIXes we claim nearly the same security criterion, only "two" must then always be replaced by "k+1".

Demonstrations of these criteria follow in the two following sections for some of the different modes and coordination protocols.

3.1.2.3.3 Skip as few as possible

We begin with an example how an attacker might bridge two MIXes if the mode "skip as few as possible" (cf. 3.1.2.3) is used and the MIXes are not properly coordinated never to apply the same transformation to the same message:

Let be MIX_u and MIX_{u+1} two consecutive MIXes, which are uncontrolled by an attacker comprising all other MIXes. If MIX_u and MIX_{u+1} act uncoordinated, the attacker may bridge them by the following trick: it submits M_u to MIX_u telling it that MIX_{u+1} is down. MIX_u will then send M_{u+2} to MIX_{u+2}. Additionally, the attacker submits M_{u+1} to MIX_{u+1} telling it that MIX_{u+2} is up (please note that our attacker need not lie that MIX_u is down, since MIX_{u+1} has no information which MIX should be its predecessor). MIX_{u+1} will then send M_{u+2} to MIX_{u+2}. The message which MIX_{u+2} receives twice is the message corresponding to M_u and M_{u+1}.

Of course the inefficient coordination protocol of 3.1.2.1 may be used. In this case a MIX which skips another MIX must take into account that it performs not one, but two transformations and execute the coordination protocol for them one after the other.

A more efficient way of coordination is given by the following efficient coordination protocol for "skip as few as possible":

After mixing and outputting the messages of a batch, each MIX sends the corresponding input messages to all other MIXes to enable them to avoid mixing these messages a second time in a situation where it is down.

A MIX only skips (= bypasses) the next MIX, if it can verify that the next MIX is really down (e.g. it can communicate with and authenticate a front end processor of the next MIX) or if a majority of all MIXes declares it to be down and there is no response from that MIX on inquiries sent regularly to it.

A MIX which is really down or declared to be down is only skipped concerning messages which it never announced to have mixed.

On the other hand, a MIX only mixes if it is in regular contact with a majority of all other MIXes. MIXes which the MIX under consideration is able to verify as down (e.g. it can communicate with and authenticate a front end processor of that MIX) need not be considered in determining a majority.

When a MIX becomes operational again or is no longer isolated from the bulk of MIXes due to a communication system outage or a denial of service attack, it first announces its ability to restart mixing and waits a certain time which allows all MIXes with which it is able to communicate to reply. It only restarts mixing after it received a confirmation of a majority of MIXes that they got notice that it intends to restart mixing. This confirmation includes a list of all messages mixed in the meantime.

If all users agree not to possibly enable every MIX to skip every other MIX, the communication necessary for coordination may be drastically reduced: each MIX sends its input messages only to those MIXes which may potentially get enabled by users to skip it. Only these other MIXes and it itself are considered

in determining a majority concerning its skipping by the others.

We think that the risk that an attacker can bridge a MIX concerning the messages for which that MIX did not get receipts yet by isolating it from the other MIXes is far outweighted by the avoidance of any additional delay.
For this coordination protocol the fact that each MIX can check that its predecessor did not tamper with the message is necessary. Otherwise an attacker could bridge two consecutive MIXes in the following way:

Let again MIX_u and MIX_{u+1} be two MIXes uncontrolled by the attacker, who controls MIX_{u-1} and MIX_{u+2}, and let X be a MIX which is down.

The attacker changes A_{u+1} in M_u to the address of X. Now MIX_u thinks that the following MIX is X, checks that X is down and consequently skips one MIX, that is MIX_u sends M_{u+2} to MIX_{u+2}.

The attacker can also form M_{u+1}, which he submits to MIX_{u+1}. So MIX_{u+1} also outputs M_{u+2} and the attacker knows that the message which MIX_{u+2} receives twice is the message corresponding to M_u.

We will now demonstrate the security criterion claimed in 3.1.2.3.2.

a) If the first MIX which receives the message, mixes, and outputs it is uncontrolled by the attacker, it provides the same security as any MIX in the original [Chau_81] scheme.

So we only have to consider the case that at least two consecutive MIXes, say MIX_u and MIX_{u+1}, are not controlled by the attacker A.

b) Let A control all other MIXes and let MIX_u and MIX_{u+1} be able to communicate (if both are up).

A has three possibilities: he may send M_u to MIX_u or he may send M_{u+1} to MIX_{u+1} telling it that MIX_{u+2} is up or down. If A does anything else, MIX_u and MIX_{u+1} will refuse to cooperate with A (please note that all schemes were constructed so that A cannot secretly tamper with messages).

Now let us consider these cases in turn.

Let us first assume that MIX_u and MIX_{u+1} are up.

C1 MIX_u receives M_u and outputs M_{u+1}. This gives no information concerning the message under consideration to the attacker, since MIX_{u-1}, which he controls, could calculate M_{u+1} as well.

Please note that this may be a threat to the unobservability of other messages: If an attacker submits n messages to a MIX and that MIX uses only n+1 messages in its batch, he can bridge the MIX with respect to all n+1 messages. This is not caused by our fault tolerance mechanisms and can be avoided by mixing messages sent by many distinct MIXes.

C2 MIX_{u+1} receives M_{u+1} and outputs M_{u+2}. This gives no information concerning the message under consideration to the attacker, since MIX_{u+1} awaits other messages as well and outputs them with changed encoding in different order.

C3 MIX_{u+1} receives M_{u+1} and outputs M_{u+3}. This gives no information concerning the message under consideration to the attacker, since MIX_{u+1} awaits other messages as well and outputs them with changed encoding in different order.

Note that it is essential that in any case MIX_u, which knows that MIX_{u+1} is up, never outputs M_{u+2}.

If one of the two MIXes is down, the other provides security by transforming M_{u+1} to M_{u+2}. The coordination protocol guarantees that the other one never performs the same transformation.

If both MIXes are down, the scheme is secure, because the transformation M_{u+1} to M_{u+2} is not performed at all. If one or both MIXes get up again, the coordination protocol guarantees that one of the cases above applies.

c) The other situation we have to consider is that MIX_u and MIX_{u+1} are not able to communicate and a majority of all MIXes is not controlled by the attacker A.

In this situation, the coordination protocol guarantees that at most one of both MIXes is declared to be up and that the one not declared up notices this after a short while (at the latest) even in case of an isolation attack mounted by A.

For all messages not mixed in this short while, the arguments for the situation where MIX_u and MIX_{u+1} could communicate (under the assumption that both are up) apply.
The number of messages mixed in this short while can be kept reasonably small, in case the efficient coordination protocol, described above, is used. If the inefficient coordination protocol (cf. section 3.1.2.1) is used, the number of messages mixed in this short while is zero.

Since all cases are exhausted, our demonstration is finished.

Since the structure of encipherment is exactly the same for the fault tolerant return address scheme as that of the fault tolerant message scheme (except for the irrelevant fact that the message part is now encrypted instead of decrypted), for which we demonstrated the security, and consecutive MIXes may be coordinated with the same protocol, the beforementioned demonstration of security applies to the security criterion for the fault tolerant untraceable return address scheme as well.

Also the demonstration of the security criterion for the schemes where each MIX can skip k other MIXes seems to be a straightforward generalization of that for the case k=1. It is omitted, since we aim at insight and not at completeness.

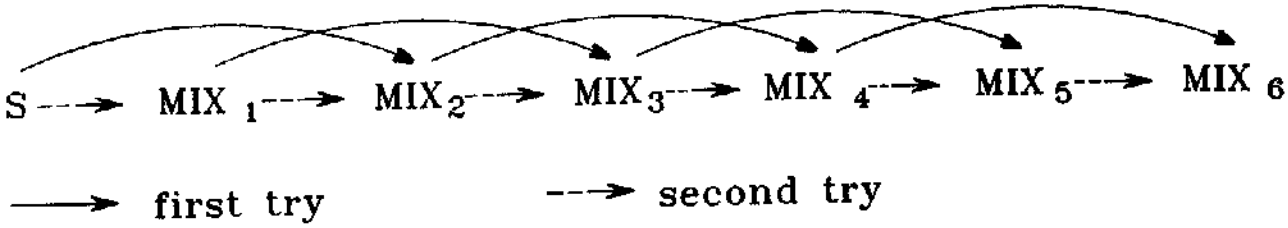
3.1.2.3.4 Skip as many as possible

To show the security of our fault tolerant schemes, it was not necessary that all k+1 consecutive uncontrolled MIXes are really passed by the message. Some may be down or as many as possible intentionally left out to (possibly) increase performance:

Whenever MIX_{i+k+1} is up, MIX_i sends M_{i+k+1} to it. Only if MIX_{i+k+1} is down, MIX_i sends M_{i+k} to MIX_{i+k}. Only if MIX_{i+k} is down, too, MIX_i sends M_{i+k-1} to MIX_{i+k-1}, and so on.

Figure 13 gives an example for k=1, using the same scenario as figure 12.

general scheme:



example: MIX 2 and MIX 4 are down

Fig. 13: Skip 1 MIX scheme operated in the mode "skip as many as possible"

Compared with the mode of operation "skip as few as possible", where MIX_i sends M_{i+1} to MIX_{i+1} if MIX_{i+1} is up, and so on, this one (possibly) saves transmission cost, (possibly) increases throughput and (possibly) decreases transfer delay in the average. The many "possibly" stem from the fact that coordination is more difficult for "skip as many as possible". Especially we cannot use our efficient coordination protocol for "skip as few as possible", since the MIX receiving a message does not get enough information to determine locally whether the mixing of a particular message replaces the action of another MIX.
If we change our schemes a little bit, the MIX may get that information:

The original sender includes a bit telling whether the particular MIX_i is replacing the action of another MIX in the first part of each message, which is encrypted with e_i. No other MIX can tamper with it, cf. 3.1.2.3.1. If MIX_i sees that it is not replacing another MIX it can execute the efficient coordination protocol, i.e. it can mix and output the message before sending the input message to all other MIXes. Otherwise it has to execute the inefficient coordination protocol, i.e. it has to submit the input message to all other MIXes before

mixing it.

If we use the original scheme and the inefficient coordination protocol as it was first described, in order to avoid the risk that MIXes may be bridged concerning messages for which they didn't get receipts yet, the difference to "skip as few as possible" is small, because a MIX must perform the coordination for each of the MIXes it skips one after the other, i.e. MIX_i first submits M_i to all other MIXes and awaits confirmation that they never mixed it before, then M_{i+1}, and so on until M_{i+k}. But if only images of messages under a one-way-function are exchanged as explained in 3.1.2.1, MIX_i may send the images of all intermediate messages $M_i,\ldots,M_{i+k}$ in parallel to the other MIXes, thus reducing not only transmission cost but also delay. This difference stems from the fact that if the attacker now submits a message to two uncontrolled MIXes, he can still see that that image which these MIXes output twice corresponds to the message he submitted twice, but if he never sees the message corresponding to this image, this is of no use to him (so neither of the two MIXes may output this message).

If all users agree not to possibly enable every MIX to skip every other MIX, the communication necessary for coordination may be drastically reduced for "skip as many as possible" as well: each MIX sends its input messages only to those MIXes, which may potentially get enabled by users to skip it, and consequently only awaits their confirmations of receipt before outputting the complete output message. Only these other MIXes, which may potentially get enabled by users to skip a particular MIX (and it itself) are considered in determining a majority concerning the skipping of that particular MIX by the others.

3.1.2.4 Quantitative evaluation

To get a quantitative evaluation and comparison of "MIXes with backups" and the "skip MIXes schemes" at reasonable effort, we model the up and down of MIXes by the reliability (or availability) r of each MIX and the security by the probability s that a particular MIX is not controlled by the attacker and executes the protocols faithfully.
We assume that reliability and security of MIXes are statistically independent from each other as well as between MIXes and that no MIX is used twice in a message (not even as backup MIX). So we may apply the usual formulas of reliability theory.
We assume that all MIXes may verify each other as down, so we need not be concerned with majorities of MIXes, as mentioned in the coordination protocols. Otherwise not only the security, but also the reliability (or availability) of our fault tolerant MIX-networks would decrease, but at least the reliability (or availability) only slightly for reasonable values of r, which are near 1.
Also, at places where this makes a difference, we will mainly consider messages in the return address scheme, because the MIX replacement schemes are particularly necessary for this case.

"MIXes with backups" are a series-parallel system. The reliability (or availability) of a system of $m \cdot k$ MIXes chosen for a message, where m MIXes are passed and each passed MIX shares its key with k-1 other MIXes, is given by

$$R = (1 - (1-r)^k)^m,$$

and its security by

$$S = 1 - (1 - s^k)^m.$$

This results from the fact that only one MIX out of each team of k MIXes must be up to enable a transmission of a message, whereas only one unsecure MIX in each team of k MIXes may undermine the anonymity of the transmission.

Using the same assumptions and notation, we will now calculate the reliability (or availability) of a system of m MIXes where each MIX may skip the next k-1 MIXes (or in other words, it may

bridge a distance of k) and compare it with the previous one.
We consider these two systems to be of the same "size", i.e. comparable, because in both of them a message passes the same number m of MIXes, yielding the same delay and transmission expense. The alternative of comparing two systems which both contain m*k MIXes because of the cost of establishing the system is unreasonable, since our performance scenarios (cf. section 2.2.3) suggest that each message may only use relatively few MIXes and hence the m*k MIXes are only a minor fraction of all MIXes in a big network.
Also note that we changed the meaning of k: in this section k gives the distance which may be bridged by a MIX, giving k the same meaning as for the scheme "MIXes with backups"; in section 3.1.2.3 k gave the number of MIXes which may be skipped (= bypassed), since that eased notation there.

We denote by $R_k(i)$ the reliability (availability) of a (sub)system of i MIXes, i.e. the probability that there is a sequence of MIXes which are up along which the message may be passed from a sender in front of the (sub)system to a recipient behind it. (The sender and recipient are assumed as up). Then

$$R_k(0) = R_k(1) = \ldots = R_k(k-1) = 1$$

because the sender can skip up to k-1 MIXes. For larger i the reliability (or availability) may be calculated recurrently by

$$R_k(i) = \sum_{j=0}^{k-1} (1-r)^j \cdot r \cdot R_k(i-j-1) \qquad \text{for } i \geq k$$

Here the j-th term is the probability that the first j of the i MIXes are down, but the j+1-st is up, and that there is a sequence of MIXes which are up along which the message may be passed among the remaining i-j-1 MIXes.
Please note that we assume that the message was constructed using the return address scheme, so this formula also holds for the complete system, because also the sender can only skip k-1

MIXes. Otherwise we would have to assume that the first MIX is always up (as long as any MIX is up in the system), because the sender could choose it arbitrarily.

The same formulas can also be used to calculate the insecurity $U_k(i)$, that is the probability that a message can be traced over a (sub)system of i MIXes from an insecure sender (i.e. under control of the attacker) directly in front of the MIXes to an insecure recipient directly behind them, starting from the insecurity u=1-s of one MIX. So we have

$$U_k(0) = U_k(1) = \ldots = U_k(k-1) = 1$$

because the attacker can bridge up to k-1 MIXes and

$$U_k(i) = \sum_{j=0}^{k-1} (1-u)^j \cdot u \cdot U_k(i-j-1) \qquad \text{for } i \geq k.$$

Here the j-th term is the probability that the first j MIXes are not controlled by the attacker, but the j+1-st is, and that there is a sequence of MIXes controlled by the attacker along which the message may be traced through the remaining i-j-1 MIXes.
$U_k(m)$ is the insecurity of the complete system only for the case of a return message the sender of which is under control of the attacker. If the sender of any message is not under control of the attacker, he will not pass the message to the first MIX under control of the attacker, but to the first MIX which is up. If this is secure, it already establishes security. So in this case we would get an insecurity $U_k'(m)$ of the complete system of

$$U_k'(m) = \sum_{j=0}^{k-1} (1-r)^j \cdot r \cdot u \cdot U_k(m-j-1).$$

(And if the sender of a normal message is under control of the attacker, there is no security anyway.)

So for the most important case of a return message the sender of which is controlled by the attacker, the security $S_k(i)$ of a (sub)system of i MIXes can be calculated by

$$S_k(0) = S_k(1) = \ldots = S_k(k-1) = 0;$$

$$S_k(i) = 1 - \sum_{j=0}^{k-1} s^j \cdot (1-s) \cdot (1 - S_k(i-j-1)) \qquad \text{for } i \geq k$$

These formulas for $R_k(m)$ and $S_k(m)$ are implemented by the following PASCAL program, written by our student Holger Bürk.
For more formulas for such consecutive-k-out-of-n:F systems, see e.g. [Shan_82].

The output of the program (generated by running it on a SIEMENS 7561 using 64 bits to represent real numbers) for some representative values of r, s, m, and k is given thereafter.
By studying this output we see that for the same values of m and k, the MIX-network using a skip MIXes scheme is more reliable (or available) but less secure than one using MIXes with backups. The difference concerning security seems to be somewhat greater, so the MIX-network using MIXes with backups seems to be favorable. But since the expense and performance of a MIX-network using a skip MIXes scheme or MIXes with backups may be quite different depending on the communication system used, the traffic to be served and the special coordination protocol applied, only an analysis taking into account all these parameters (and fixing at least some of them) can decide definitly.

```
program evaluation (input,output);

{ This program calculates a table of values for

    r_net   (= reliability of the fault tolerant MIX-network
               using skip MIXes schemes )
    s_net   (= security of the fault tolerant MIX-network
               using skip MIXes schemes )
    r_comp  (= reliability of the MIX-network using MIXes
               with backups forming a series-parallel system )
    s_comp  (= security of the MIX-network using MIXes
               with backups forming a series-parallel system )

  which reaches for k from 1 to k_max and for m from 1 to m_max

  Input for this program are the values for:

    r       (= reliability for one MIX)
    s       (= security for one MIX)
    m_max   (= maximum for the number of MIXes passed)
    k_max   (= maximum for the distance between MIXes which can
               be bridged, number of parallel MIXes respectively)

  Author    : Holger Buerk     }

const mmax = 9999999;{ upper bound for m }
      kmax = 7;       { upper bound for k }
      lmax = 700;     { upper bound for number of stored results }

type result = record r    : real;
                     s    : real;
                     rcom : real;
                     scom : real;
                     m    : integer;
              end;
     matrix = array [1..lmax,1..kmax] of result;
     vector = array [0..kmax] of real;
var
  hr    : vector; { vector for calculation of the
                    recurrence-relation for reliability        }
  hs    : vector; { as hr, but for security                      }
  res   : matrix; { matrix of the results                        }
  s     : real;   { security of one MIX                          }
  r     : real;   { reliability of one MIX                       }
  s_net : real;   { security of the MIX-network using
                    skip MIXes schemes                         }
  r_net : real;   { reliability for the MIX-network using
                    skip MIXes schemes                         }
  s_comp: real;   { security of the MIX-network using MIXes with
                    backups forming a series-parallel system   }
  r_comp: real;   { reliability of the MIX-network using MIXes
                    with backups forming a series-parallel
                    system                                     }
  m     : integer;{ current number of MIXes passed             }
  k     : integer;{ current distance between MIXes which can be
                    bridged, or number of parallel MIXes       }
  m_max : integer;{ max. number of MIXes for which results are
```

```
                    calculated                                          }
  k_max : integer;{ max. distance between MIXes which can be
                    bridged for which results are calculated          }
  m_old : integer;{ marker for m, for which the last result was
                    calculated                                        }
  k_old : integer;{ marker for k, for which the last result was
                    calculated                                        }
  l     : integer;{ vertical index of matrix 'res'                    }
  l_max : integer;{ last vertical index of matrix 'res'               }

procedure calc (m,k:integer;var l:integer;var res:matrix);

{ This procedure fills up one field of the resulting matrix
  'res', i.e. res[l,k], with corresponding values. For the
  calculation of these values the procedure 'calc1' will be
  used.                                                               }

var
  newstart : boolean; { controls whether already calculated
                        values will be used or not                    }
procedure calc1 (m,k:integer;newstart:boolean;p:real;
                 var h : vector;var res: real);

{ This procedure calculates values of the security or
  reliability of the MIX-network using the recurrence equation.
  In the case that k has not changed since the last calculation,
  the last calculated values of the recurrence relation are
  usable.                                                             }

  var
    x : real;      { intermediate storage for the result of calc.}
    j : integer;   { index for the recurrence-relation-vector    }
    i : integer;   { marker for running n                        }
  begin
    if newstart then begin
      { calculation from beginning }
      for j := 0 to k-1 do h[j] := 1;
      i := k;
      end
    else i := m_old +1;{uses already calculated values for calc.}
    if m < k then res := 1
    else while i <= m do begin
      x := 0;
      for j := 0 to k-1 do
         x := x + exp ( (k-1-j)*ln(1-p) )*h[j];
      res := x*p;
      for j := 0 to k-2 do h[j] := h[j+1];
      h[k-1] := res;
      i := i + 1;
      end;
  end;
{ end of procedure calc1 }

begin
  if k = k_old then newstart := false else newstart := true;
  res [l,k].m := m;
  calc1 (m,k,newstart,1-s,hs,res[l,k].s);
```

```
    res[l,k].s := 1 - res[l,k].s;
    calc1 (m,k,newstart,r,hr,res[l,k].r);
    if m < k then begin
      res [l,k].scom := 0;
      res [l,k].rcom := 1;
      end
    else begin
      res[l,k].scom := 1 - exp (m*ln(1-exp(k*ln(s))));
      res[l,k].rcom := exp (m*ln(1-exp(k*ln(1-r))));
      end;
    k_old := k;
    m_old := m;
  end;
  { end of procedure calc }


  procedure init (var m_max,k_max:integer;var s,r,:real);

  { procedure where the values for
    m_max, k_max, s, r will be inserted }

    var
      q : boolean;
    begin
      writeln ('************* p a r a m e t e r s *************');
      writeln;
      q := false;
      while not q do begin
        q := true;
        write ('value     for r : ');
        read(r);writeln ('  ',r:6:6);
        write ('value     for s : ');
        read(s);writeln ('  ',s:6:6);
        write ('value for m_max : ');
        read(m_max);writeln(' ',m_max);
        write ('value for k_max : ');
        read(k_max);writeln(' ',k_max);
        if (r<0) or (r>1) or (s<0) or (s>1) then q := false;
        if (m_max>mmax) or (m_max<1) or (k_max>kmax) or (k_max<1)
        then q := false;
        end;
      writeln ('***********************************************');
    end;
  { end of procedure init }

  procedure tab_print (res:matrix;k_max,l_max:integer);

  { This procedure produces the table with the results }

    var
      j,i : integer;
    begin
      writeln;
      writeln;
      write ('The resulting table contains ');
      writeln ('in every field four values');
      writeln;
```

```
    write ('  s_net  r_net  for security/reliability ');
    writeln ('of the MIX-network');
    write ('  s_com  r_com  as above ');
    writeln ('but of a series-parallel circuit');
    writeln;
    write ('the horizontal index is for values of ''k''; ');
    write ('the vertical');
    writeln (' index is for values of ''m''');
    writeln;
    writeln;
    write ('          ');
    for j := 1 to k_max do write ('        ',j:1,'        ');
    writeln;
    for j := 1 to (17*k_max + 9) do write ('=');
    writeln;
    for i := 1 to l_max do begin
      write (res[i,1].m:7,'  ');
      for j := 1 to k_max do
        write ('  ',res[i,j].s:5:5,' ',res[i,j].r:5:5);
      writeln;
      write ('          ');
      for j := 1 to k_max do
        write ('  ',res[i,j].scom:5:5,' ',res[i,j].rcom:5:5);
      writeln;
      for j := 1 to (17*k_max + 9) do write ('-');
      writeln;
      end;
 end;
{ end of procedure tab_print }


{  M A I N   S E C T I O N  }
begin
  init (m_max,k_max,s,r);
  m := 1;
  k := 1;
  m_old := 0;
  k_old := 0;
  l := 1;
  while k <= k_max do begin
    l_max := l;
    if m > m_max then begin
      k := k + 1;
      m := 1;
      l := 1;
      m_old := 0;
      end
    else begin
      calc (m,k,l,res);
      if m < 100 then m := m+1
      else m := m + 1 +
                trunc( exp(ln(10)*trunc(ln(m)/ln(10)- 1)) );
      l := l+1;
      end;
    end;
  tab_print (res,k_max,l_max-1);
end.
```

********** p a r a m e t e r s **********

value for r : 0.990000
value for s : 0.900000
value for m_max : 50
value for k_max : 5
****

The resulting table contains in every field four values

s_net r_net for security/reliability of the MIX-network
s_com r_com as above but of a series-parallel circuit

the horizontal index is for values of 'k'; the vertical index is for values of 'm'

	1	2	3	4	5
1	0.90000 0.99000 0.90000 0.99000	0.00000 1.00000 0.00000 1.00000	0.00000 1.00000 0.00000 1.00000	0.00000 1.00000 0.00000 1.00000	0.00000 1.00000 0.00000 1.00000
2	0.99000 0.98010 0.99000 0.98010	0.81000 0.99990 0.96390 0.99980	0.00000 1.00000 0.00000 1.00000	0.00000 1.00000 0.00000 1.00000	0.00000 1.00000 0.00000 1.00000
3	0.99900 0.97030 0.99900 0.97030	0.89100 0.99980 0.99314 0.99970	0.72900 1.00000 0.98010 1.00000	0.00000 1.00000 0.00000 1.00000	0.00000 1.00000 0.00000 1.00000
4	0.99990 0.96060 0.99990 0.96060	0.97200 0.99970 0.99870 0.99960	0.80190 1.00000 0.99461 1.00000	0.65610 1.00000 0.98601 1.00000	0.00000 1.00000 0.00000 1.00000
5	0.99999 0.95099 0.99999 0.95099	0.98739 0.99960 0.99975 0.99950	0.87480 1.00000 0.99854 1.00000	0.72171 1.00000 0.99519 1.00000	0.59049 1.00000 0.98848 1.00000
6	1.00000 0.94148 1.00000 0.94148	0.99622 0.99950 0.99995 0.99940	0.94770 1.00000 0.99960 0.99999	0.78732 1.00000 0.99835 1.00000	0.64954 1.00000 0.99528 1.00000
7	1.00000 0.93207 1.00000 0.93207	0.99849 0.99941 0.99999 0.99930	0.96746 1.00000 0.99989 0.99999	0.85293 1.00000 0.99943 1.00000	0.70859 1.00000 0.99807 1.00000
8	1.00000 0.92274 1.00000 0.92274	0.99951 0.99931 1.00000 0.99920	0.98190 0.99999 0.99997 0.99999	0.91854 1.00000 0.99980 1.00000	0.76764 1.00000 0.99921 1.00000
9	1.00000 0.91352 1.00000 0.91352	0.99981 0.99921 1.00000 0.99910	0.99102 0.99999 0.99999 0.99999	0.94110 1.00000 0.99993 1.00000	0.82669 1.00000 0.99968 1.00000
10	1.00000 0.90438 1.00000 0.90438	0.99994 0.99911 1.00000 0.99900	0.99484 0.99999 1.00000 0.99999	0.95936 1.00000 0.99998 1.00000	0.88573 1.00000 0.99987 1.00000
11	1.00000 0.89534 1.00000 0.89534	0.99998 0.99901 1.00000 0.99890	0.99721 0.99999 1.00000 0.99999	0.97332 1.00000 0.99999 1.00000	0.90992 1.00000 0.99995 1.00000
12	1.00000 0.88638 1.00000 0.88638	0.99999 0.99891 1.00000 0.99880	0.99853 0.99999 1.00000 0.99999	0.98297 1.00000 1.00000 1.00000	0.93061 1.00000 0.99998 1.00000
13	1.00000 0.87752 1.00000 0.87752	1.00000 0.99881 1.00000 0.99870	0.99918 0.99999 1.00000 0.99999	0.98831 1.00000 1.00000 1.00000	0.94782 1.00000 0.99999 1.00000
14	1.00000 0.86875 1.00000 0.86875	1.00000 0.99871 1.00000 0.99860	0.99956 0.99999 1.00000 0.99999	0.99217 1.00000 1.00000 1.00000	0.96154 1.00000 1.00000 1.00000
15	1.00000 0.86006 1.00000 0.86006	1.00000 0.99861 1.00000 0.99850	0.99976 0.99999 1.00000 0.99999	0.99484 1.00000 1.00000 1.00000	0.97177 1.00000 1.00000 1.00000
16	1.00000 0.85146 1.00000 0.85146	1.00000 0.99851 1.00000 0.99840	0.99987 0.99999 1.00000 0.99998	0.99659 1.00000 1.00000 1.00000	0.97852 1.00000 1.00000 1.00000
17	1.00000 0.84294 1.00000 0.84294	1.00000 0.99842 1.00000 0.99830	0.99993 0.99999 1.00000 0.99998	0.99771 1.00000 1.00000 1.00000	0.98384 1.00000 1.00000 1.00000
18	1.00000 0.83451 1.00000 0.83451	1.00000 0.99832 1.00000 0.99820	0.99996 0.99998 1.00000 0.99998	0.99848 1.00000 1.00000 1.00000	0.98794 1.00000 1.00000 1.00000
19	1.00000 0.82617 1.00000 0.82617	1.00000 0.99822 1.00000 0.99810	0.99998 0.99998 1.00000 0.99998	0.99899 1.00000 1.00000 1.00000	0.99102 1.00000 1.00000 1.00000
20	1.00000 0.81791 1.00000 0.81791	1.00000 0.99812 1.00000 0.99800	0.99999 0.99998 1.00000 0.99998	0.99933 1.00000 1.00000 1.00000	0.99329 1.00000 1.00000 1.00000
21	1.00000 0.80973 1.00000 0.80973	1.00000 0.99802 1.00000 0.99790	0.99999 0.99998 1.00000 0.99998	0.99955 1.00000 1.00000 1.00000	0.99496 1.00000 1.00000 1.00000
22	1.00000 0.80163 1.00000 0.80163	1.00000 0.99792 1.00000 0.99780	1.00000 0.99998 1.00000 0.99998	0.99970 1.00000 1.00000 1.00000	0.99622 1.00000 1.00000 1.00000
23	1.00000 0.79361 1.00000 0.79361	1.00000 0.99782 1.00000 0.99770	1.00000 0.99998 1.00000 0.99998	0.99980 1.00000 1.00000 1.00000	0.99718 1.00000 1.00000 1.00000
24	1.00000 0.78568 1.00000 0.78568	1.00000 0.99772 1.00000 0.99760	1.00000 0.99998 1.00000 0.99998	0.99987 1.00000 1.00000 1.00000	0.99789 1.00000 1.00000 1.00000
25	1.00000 0.77782 1.00000 0.77782	1.00000 0.99763 1.00000 0.99750	1.00000 0.99998 1.00000 0.99998	0.99991 1.00000 1.00000 1.00000	0.99842 1.00000 1.00000 1.00000
26	1.00000 0.77004 1.00000 0.77004	1.00000 0.99753 1.00000 0.99740	1.00000 0.99998 1.00000 0.99997	0.99994 1.00000 1.00000 1.00000	0.99882 1.00000 1.00000 1.00000

27	1.00000 0.76234	1.00000 0.99743	1.00000 0.99998	0.99996 1.00000	0.99912 1.00000
	1.00000 0.76234	1.00000 0.99730	1.00000 0.99997	1.00000 1.00000	1.00000 1.00000
28	1.00000 0.75472	1.00000 0.99733	1.00000 0.99997	0.99997 1.00000	0.99934 1.00000
	1.00000 0.75472	1.00000 0.99720	1.00000 0.99997	1.00000 1.00000	1.00000 1.00000
29	1.00000 0.74717	1.00000 0.99723	1.00000 0.99997	0.99998 1.00000	0.99951 1.00000
	1.00000 0.74717	1.00000 0.99710	1.00000 0.99997	1.00000 1.00000	1.00000 1.00000
30	1.00000 0.73970	1.00000 0.99713	1.00000 0.99997	0.99999 1.00000	0.99963 1.00000
	1.00000 0.73970	1.00000 0.99700	1.00000 0.99997	1.00000 1.00000	1.00000 1.00000
31	1.00000 0.73230	1.00000 0.99703	1.00000 0.99997	0.99999 1.00000	0.99972 1.00000
	1.00000 0.73230	1.00000 0.99690	1.00000 0.99997	1.00000 1.00000	1.00000 1.00000
32	1.00000 0.72498	1.00000 0.99693	1.00000 0.99997	0.99999 1.00000	0.99979 1.00000
	1.00000 0.72498	1.00000 0.99680	1.00000 0.99997	1.00000 1.00000	1.00000 1.00000
33	1.00000 0.71773	1.00000 0.99684	1.00000 0.99997	1.00000 1.00000	0.99984 1.00000
	1.00000 0.71773	1.00000 0.99671	1.00000 0.99997	1.00000 1.00000	1.00000 1.00000
34	1.00000 0.71055	1.00000 0.99674	1.00000 0.99997	1.00000 1.00000	0.99988 1.00000
	1.00000 0.71055	1.00000 0.99661	1.00000 0.99997	1.00000 1.00000	1.00000 1.00000
35	1.00000 0.70345	1.00000 0.99664	1.00000 0.99997	1.00000 1.00000	0.99991 1.00000
	1.00000 0.70345	1.00000 0.99651	1.00000 0.99997	1.00000 1.00000	1.00000 1.00000
36	1.00000 0.69641	1.00000 0.99654	1.00000 0.99997	1.00000 1.00000	0.99993 1.00000
	1.00000 0.69641	1.00000 0.99641	1.00000 0.99996	1.00000 1.00000	1.00000 1.00000
37	1.00000 0.68945	1.00000 0.99644	1.00000 0.99997	1.00000 1.00000	0.99995 1.00000
	1.00000 0.68945	1.00000 0.99631	1.00000 0.99996	1.00000 1.00000	1.00000 1.00000
38	1.00000 0.68255	1.00000 0.99634	1.00000 0.99996	1.00000 1.00000	0.99996 1.00000
	1.00000 0.68255	1.00000 0.99621	1.00000 0.99996	1.00000 1.00000	1.00000 1.00000
39	1.00000 0.67573	1.00000 0.99624	1.00000 0.99996	1.00000 1.00000	0.99997 1.00000
	1.00000 0.67573	1.00000 0.99611	1.00000 0.99996	1.00000 1.00000	1.00000 1.00000
40	1.00000 0.66897	1.00000 0.99614	1.00000 0.99996	1.00000 1.00000	0.99998 1.00000
	1.00000 0.66897	1.00000 0.99601	1.00000 0.99996	1.00000 1.00000	1.00000 1.00000
41	1.00000 0.66228	1.00000 0.99605	1.00000 0.99996	1.00000 1.00000	0.99998 1.00000
	1.00000 0.66228	1.00000 0.99591	1.00000 0.99996	1.00000 1.00000	1.00000 1.00000
42	1.00000 0.65566	1.00000 0.99595	1.00000 0.99996	1.00000 1.00000	0.99999 1.00000
	1.00000 0.65566	1.00000 0.99581	1.00000 0.99996	1.00000 1.00000	1.00000 1.00000
43	1.00000 0.64910	1.00000 0.99585	1.00000 0.99996	1.00000 1.00000	0.99999 1.00000
	1.00000 0.64910	1.00000 0.99571	1.00000 0.99996	1.00000 1.00000	1.00000 1.00000
44	1.00000 0.64261	1.00000 0.99575	1.00000 0.99996	1.00000 1.00000	0.99999 1.00000
	1.00000 0.64261	1.00000 0.99561	1.00000 0.99996	1.00000 1.00000	1.00000 1.00000
45	1.00000 0.63619	1.00000 0.99565	1.00000 0.99996	1.00000 1.00000	1.00000 1.00000
	1.00000 0.63619	1.00000 0.99551	1.00000 0.99996	1.00000 1.00000	1.00000 1.00000
46	1.00000 0.62982	1.00000 0.99555	1.00000 0.99996	1.00000 1.00000	1.00000 1.00000
	1.00000 0.62982	1.00000 0.99541	1.00000 0.99995	1.00000 1.00000	1.00000 1.00000
47	1.00000 0.62353	1.00000 0.99545	1.00000 0.99996	1.00000 1.00000	1.00000 1.00000
	1.00000 0.62353	1.00000 0.99531	1.00000 0.99995	1.00000 1.00000	1.00000 1.00000
48	1.00000 0.61729	1.00000 0.99536	1.00000 0.99995	1.00000 1.00000	1.00000 1.00000
	1.00000 0.61729	1.00000 0.99521	1.00000 0.99995	1.00000 1.00000	1.00000 1.00000
49	1.00000 0.61112	1.00000 0.99526	1.00000 0.99995	1.00000 1.00000	1.00000 1.00000
	1.00000 0.61112	1.00000 0.99511	1.00000 0.99995	1.00000 1.00000	1.00000 1.00000
50	1.00000 0.60501	1.00000 0.99516	1.00000 0.99995	1.00000 1.00000	1.00000 1.00000
	1.00000 0.60501	1.00000 0.99501	1.00000 0.99995	1.00000 1.00000	1.00000 1.00000

********** p a r a m e t e r s **********

value for r : 0.990000
value for s : 0.600000
value for m_max : 100
value for k_max : 5
****

The resulting table contains in every field four values

s_net r_net for security/reliability of the MIX-network
s_com r_com as above but of a series-parallel circuit

the horizontal index is for values of 'k'; the vertical index is for values of 'm'

	1	2	3	4	5
1	0.60000 0.99000 0.60000 0.99000	0.00000 1.00000 0.00000 1.00000	0.00000 1.00000 0.00000 1.00000	0.00000 1.00000 0.00000 1.00000	0.00000 1.00000 0.00000 1.00000
2	0.84000 0.98010 0.84000 0.98010	0.36000 0.99990 0.59040 0.99980	0.00000 1.00000 0.00000 1.00000	0.00000 1.00000 0.00000 1.00000	0.00000 1.00000 0.00000 1.00000
3	0.93600 0.97030 0.93600 0.97030	0.50400 0.99980 0.73786 0.99970	0.21600 1.00000 0.51811 1.00000	0.00000 1.00000 0.00000 1.00000	0.00000 1.00000 0.00000 1.00000
4	0.97440 0.96060 0.97440 0.96060	0.64800 0.99970 0.83223 0.99960	0.30240 1.00000 0.62220 1.00000	0.12960 1.00000 0.42605 1.00000	0.00000 1.00000 0.00000 1.00000
5	0.98976 0.95099 0.98976 0.95099	0.74016 0.99960 0.89263 0.99950	0.38880 1.00000 0.70380 1.00000	0.18144 1.00000 0.50043 1.00000	0.07776 1.00000 0.33286 1.00000
6	0.99590 0.94148 0.99590 0.94148	0.81158 0.99950 0.93128 0.99940	0.47520 1.00000 0.76778 0.99999	0.23328 1.00000 0.56518 1.00000	0.10886 1.00000 0.38473 1.00000
7	0.99836 0.93207 0.99836 0.93207	0.86227 0.99941 0.95602 0.99930	0.54294 1.00000 0.81794 0.99999	0.28512 1.00000 0.62153 1.00000	0.13997 1.00000 0.43258 1.00000
8	0.99934 0.92274 0.99934 0.92274	0.89969 0.99931 0.97185 0.99920	0.60321 0.99999 0.85727 0.99999	0.33696 1.00000 0.67058 1.00000	0.17107 1.00000 0.47670 1.00000
9	0.99974 0.91352 0.99974 0.91352	0.92682 0.99921 0.98199 0.99910	0.65602 0.99999 0.88810 0.99999	0.38208 1.00000 0.71327 1.00000	0.20218 1.00000 0.51739 1.00000
10	0.99990 0.90438 0.99990 0.90438	0.94665 0.99911 0.98847 0.99900	0.70136 0.99999 0.91227 0.99999	0.42452 1.00000 0.75043 1.00000	0.23328 1.00000 0.55492 1.00000
11	0.99996 0.89534 0.99996 0.89534	0.96110 0.99901 0.99262 0.99890	0.74085 0.99999 0.93122 0.99999	0.46426 1.00000 0.78278 1.00000	0.26197 1.00000 0.58953 1.00000
12	0.99998 0.88638 0.99998 0.88638	0.97164 0.99891 0.99528 0.99880	0.77513 0.99999 0.94607 0.99999	0.50132 1.00000 0.81093 1.00000	0.28968 1.00000 0.62145 1.00000
13	0.99999 0.87752 0.99999 0.87752	0.97932 0.99881 0.99698 0.99870	0.80485 0.99999 0.95772 0.99999	0.53569 1.00000 0.83543 1.00000	0.31643 1.00000 0.65088 1.00000
14	1.00000 0.86875 1.00000 0.86875	0.98492 0.99871 0.99807 0.99860	0.83066 0.99999 0.96685 0.99999	0.56773 1.00000 0.85676 1.00000	0.34222 1.00000 0.67803 1.00000
15	1.00000 0.86006 1.00000 0.86006	0.98900 0.99861 0.99876 0.99850	0.85305 0.99999 0.97401 0.99999	0.59756 1.00000 0.87532 1.00000	0.36703 1.00000 0.70307 1.00000
16	1.00000 0.85146 1.00000 0.85146	0.99198 0.99851 0.99921 0.99840	0.87247 0.99999 0.97963 0.99998	0.62533 1.00000 0.89148 1.00000	0.39088 1.00000 0.72616 1.00000
17	1.00000 0.84294 1.00000 0.84294	0.99415 0.99842 0.99949 0.99830	0.88934 0.99999 0.98403 0.99998	0.65118 1.00000 0.90555 1.00000	0.41384 1.00000 0.74745 1.00000
18	1.00000 0.83451 1.00000 0.83451	0.99574 0.99832 0.99968 0.99820	0.90397 0.99998 0.98748 0.99998	0.67525 1.00000 0.91779 1.00000	0.43593 1.00000 0.76709 1.00000
19	1.00000 0.82617 1.00000 0.82617	0.99689 0.99822 0.99979 0.99810	0.91666 0.99998 0.99018 0.99998	0.69766 1.00000 0.92844 1.00000	0.45719 1.00000 0.78520 1.00000
20	1.00000 0.81791 1.00000 0.81791	0.99773 0.99812 0.99987 0.99800	0.92768 0.99998 0.99230 0.99998	0.71853 1.00000 0.93772 1.00000	0.47765 1.00000 0.80190 1.00000
21	1.00000 0.80973 1.00000 0.80973	0.99835 0.99802 0.99991 0.99790	0.93724 0.99998 0.99397 0.99998	0.73795 1.00000 0.94579 1.00000	0.49734 1.00000 0.81731 1.00000
22	1.00000 0.80163 1.00000 0.80163	0.99880 0.99792 0.99995 0.99780	0.94554 0.99998 0.99527 0.99998	0.75603 1.00000 0.95281 1.00000	0.51628 1.00000 0.83151 1.00000
23	1.00000 0.79361 1.00000 0.79361	0.99912 0.99782 0.99997 0.99770	0.95274 0.99998 0.99629 0.99998	0.77287 1.00000 0.95893 1.00000	0.53452 1.00000 0.84461 1.00000
24	1.00000 0.78568 1.00000 0.78568	0.99936 0.99772 0.99998 0.99760	0.95899 0.99998 0.99709 0.99998	0.78854 1.00000 0.96425 1.00000	0.55206 1.00000 0.85670 1.00000
25	1.00000 0.77782 1.00000 0.77782	0.99953 0.99763 0.99999 0.99750	0.96441 0.99998 0.99772 0.99998	0.80313 1.00000 0.96888 1.00000	0.56895 1.00000 0.86784 1.00000
26	1.00000 0.77004 1.00000 0.77004	0.99966 0.99753 0.99999 0.99740	0.96912 0.99998 0.99821 0.99997	0.81671 1.00000 0.97292 1.00000	0.58519 1.00000 0.87812 1.00000

27	1.00000 0.76234	0.99975 0.99743	0.97320 0.99998	0.82936 1.00000	0.60083 1.00000
	1.00000 0.76234	0.99999 0.99730	0.99860 0.99997	0.97643 1.00000	0.88759 1.00000
28	1.00000 0.75472	0.99982 0.99733	0.97674 0.99997	0.84114 1.00000	0.61587 1.00000
	1.00000 0.75472	1.00000 0.99720	0.99890 0.99997	0.97948 1.00000	0.89634 1.00000
29	1.00000 0.74717	0.99987 0.99723	0.97982 0.99997	0.85210 1.00000	0.63035 1.00000
	1.00000 0.74717	1.00000 0.99710	0.99914 0.99997	0.98214 1.00000	0.90440 1.00000
30	1.00000 0.73970	0.99990 0.99713	0.98249 0.99997	0.86230 1.00000	0.64428 1.00000
	1.00000 0.73970	1.00000 0.99700	0.99932 0.99997	0.98446 1.00000	0.91183 1.00000
31	1.00000 0.73230	0.99993 0.99703	0.98480 0.99997	0.87181 1.00000	0.65769 1.00000
	1.00000 0.73230	1.00000 0.99690	0.99947 0.99997	0.98647 1.00000	0.91869 1.00000
32	1.00000 0.72498	0.99995 0.99693	0.98681 0.99997	0.88065 1.00000	0.67059 1.00000
	1.00000 0.72498	1.00000 0.99680	0.99958 0.99997	0.98822 1.00000	0.92501 1.00000
33	1.00000 0.71773	0.99996 0.99684	0.98855 0.99997	0.88889 1.00000	0.68301 1.00000
	1.00000 0.71773	1.00000 0.99671	0.99967 0.99997	0.98975 1.00000	0.93084 1.00000
34	1.00000 0.71055	0.99997 0.99674	0.99007 0.99997	0.89655 1.00000	0.69496 1.00000
	1.00000 0.71055	1.00000 0.99661	0.99974 0.99997	0.99108 1.00000	0.93622 1.00000
35	1.00000 0.70345	0.99998 0.99664	0.99138 0.99997	0.90369 1.00000	0.70645 1.00000
	1.00000 0.70345	1.00000 0.99651	0.99980 0.99997	0.99223 1.00000	0.94118 1.00000
36	1.00000 0.69641	0.99999 0.99654	0.99252 0.99997	0.91034 1.00000	0.71752 1.00000
	1.00000 0.69641	1.00000 0.99641	0.99984 0.99996	0.99324 1.00000	0.94575 1.00000
37	1.00000 0.68945	0.99999 0.99644	0.99351 0.99997	0.91653 1.00000	0.72817 1.00000
	1.00000 0.68945	1.00000 0.99631	0.99988 0.99996	0.99412 1.00000	0.94997 1.00000
38	1.00000 0.68255	0.99999 0.99634	0.99437 0.99996	0.92229 1.00000	0.73841 1.00000
	1.00000 0.68255	1.00000 0.99621	0.99990 0.99996	0.99488 1.00000	0.95386 1.00000
39	1.00000 0.67573	0.99999 0.99624	0.99511 0.99996	0.92765 1.00000	0.74827 1.00000
	1.00000 0.67573	1.00000 0.99611	0.99992 0.99996	0.99554 1.00000	0.95745 1.00000
40	1.00000 0.66897	1.00000 0.99614	0.99576 0.99996	0.93264 1.00000	0.75776 1.00000
	1.00000 0.66897	1.00000 0.99601	0.99994 0.99996	0.99612 1.00000	0.96076 1.00000
41	1.00000 0.66228	1.00000 0.99605	0.99632 0.99996	0.93729 1.00000	0.76689 1.00000
	1.00000 0.66228	1.00000 0.99591	0.99995 0.99996	0.99662 1.00000	0.96381 1.00000
42	1.00000 0.65566	1.00000 0.99595	0.99681 0.99996	0.94162 1.00000	0.77568 1.00000
	1.00000 0.65566	1.00000 0.99581	0.99996 0.99996	0.99706 1.00000	0.96662 1.00000
43	1.00000 0.64910	1.00000 0.99585	0.99723 0.99996	0.94564 1.00000	0.78413 1.00000
	1.00000 0.64910	1.00000 0.99571	0.99997 0.99996	0.99744 1.00000	0.96922 1.00000
44	1.00000 0.64261	1.00000 0.99575	0.99759 0.99996	0.94940 1.00000	0.79227 1.00000
	1.00000 0.64261	1.00000 0.99561	0.99998 0.99996	0.99777 1.00000	0.97161 1.00000
45	1.00000 0.63619	1.00000 0.99565	0.99791 0.99996	0.95289 1.00000	0.80010 1.00000
	1.00000 0.63619	1.00000 0.99551	0.99998 0.99996	0.99806 1.00000	0.97382 1.00000
46	1.00000 0.62982	1.00000 0.99555	0.99819 0.99996	0.95614 1.00000	0.80763 1.00000
	1.00000 0.62982	1.00000 0.99541	0.99999 0.99995	0.99831 1.00000	0.97586 1.00000
47	1.00000 0.62353	1.00000 0.99545	0.99843 0.99996	0.95917 1.00000	0.81488 1.00000
	1.00000 0.62353	1.00000 0.99531	0.99999 0.99995	0.99853 1.00000	0.97773 1.00000
48	1.00000 0.61729	1.00000 0.99536	0.99864 0.99995	0.96198 1.00000	0.82186 1.00000
	1.00000 0.61729	1.00000 0.99521	0.99999 0.99995	0.99872 1.00000	0.97946 1.00000
49	1.00000 0.61112	1.00000 0.99526	0.99882 0.99995	0.96461 1.00000	0.82857 1.00000
	1.00000 0.61112	1.00000 0.99511	0.99999 0.99995	0.99889 1.00000	0.98106 1.00000
50	1.00000 0.60501	1.00000 0.99516	0.99897 0.99995	0.96705 1.00000	0.83504 1.00000
	1.00000 0.60501	1.00000 0.99501	0.99999 0.99995	0.99903 1.00000	0.98253 1.00000
51	1.00000 0.59896	1.00000 0.99506	0.99911 0.99995	0.96932 1.00000	0.84125 1.00000
	1.00000 0.59896	1.00000 0.99491	1.00000 0.99995	0.99916 1.00000	0.98389 1.00000
52	1.00000 0.59297	1.00000 0.99496	0.99923 0.99995	0.97144 1.00000	0.84724 1.00000
	1.00000 0.59297	1.00000 0.99481	1.00000 0.99995	0.99927 1.00000	0.98514 1.00000
53	1.00000 0.58704	1.00000 0.99486	0.99933 0.99995	0.97341 1.00000	0.85300 1.00000
	1.00000 0.58704	1.00000 0.99471	1.00000 0.99995	0.99936 1.00000	0.98630 1.00000
54	1.00000 0.58117	1.00000 0.99476	0.99942 0.99995	0.97524 1.00000	0.85854 1.00000
	1.00000 0.58117	1.00000 0.99461	1.00000 0.99995	0.99944 1.00000	0.98737 1.00000
55	1.00000 0.57535	1.00000 0.99467	0.99949 0.99995	0.97695 1.00000	0.86387 1.00000
	1.00000 0.57535	1.00000 0.99451	1.00000 0.99995	0.99952 1.00000	0.98835 1.00000
56	1.00000 0.56960	1.00000 0.99457	0.99956 0.99995	0.97854 1.00000	0.86900 1.00000
	1.00000 0.56960	1.00000 0.99442	1.00000 0.99994	0.99958 1.00000	0.98925 1.00000
57	1.00000 0.56391	1.00000 0.99447	0.99962 0.99995	0.98002 1.00000	0.87394 1.00000
	1.00000 0.56391	1.00000 0.99432	1.00000 0.99994	0.99963 1.00000	0.99009 1.00000
58	1.00000 0.55827	1.00000 0.99437	0.99967 0.99994	0.98140 1.00000	0.87869 1.00000
	1.00000 0.55827	1.00000 0.99422	1.00000 0.99994	0.99968 1.00000	0.99086 1.00000

59	1.00000	0.55268	1.00000	0.99427	0.99971	0.99994	0.98269	1.00000	0.88326	1.00000
	1.00000	0.55268	1.00000	0.99412	1.00000	0.99994	0.99972	1.00000	0.99157	1.00000
60	1.00000	0.54716	1.00000	0.99417	0.99975	0.99994	0.98388	1.00000	0.88766	1.00000
	1.00000	0.54716	1.00000	0.99402	1.00000	0.99994	0.99976	1.00000	0.99223	1.00000
61	1.00000	0.54169	1.00000	0.99408	0.99978	0.99994	0.98499	1.00000	0.89190	1.00000
	1.00000	0.54169	1.00000	0.99392	1.00000	0.99994	0.99979	1.00000	0.99283	1.00000
62	1.00000	0.53627	1.00000	0.99398	0.99981	0.99994	0.98603	1.00000	0.89597	1.00000
	1.00000	0.53627	1.00000	0.99382	1.00000	0.99994	0.99982	1.00000	0.99339	1.00000
63	1.00000	0.53091	1.00000	0.99388	0.99984	0.99994	0.98699	1.00000	0.89989	1.00000
	1.00000	0.53091	1.00000	0.99372	1.00000	0.99994	0.99984	1.00000	0.99390	1.00000
64	1.00000	0.52560	1.00000	0.99378	0.99986	0.99994	0.98789	1.00000	0.90366	1.00000
	1.00000	0.52560	1.00000	0.99362	1.00000	0.99994	0.99986	1.00000	0.99438	1.00000
65	1.00000	0.52034	1.00000	0.99368	0.99988	0.99994	0.98873	1.00000	0.90730	1.00000
	1.00000	0.52034	1.00000	0.99352	1.00000	0.99994	0.99988	1.00000	0.99481	1.00000
66	1.00000	0.51514	1.00000	0.99358	0.99989	0.99994	0.98950	1.00000	0.91079	1.00000
	1.00000	0.51514	1.00000	0.99342	1.00000	0.99993	0.99989	1.00000	0.99522	1.00000
67	1.00000	0.50999	1.00000	0.99348	0.99991	0.99994	0.99023	1.00000	0.91415	1.00000
	1.00000	0.50999	1.00000	0.99332	1.00000	0.99993	0.99991	1.00000	0.99559	1.00000
68	1.00000	0.50489	1.00000	0.99339	0.99992	0.99993	0.99090	1.00000	0.91739	1.00000
	1.00000	0.50489	1.00000	0.99322	1.00000	0.99993	0.99992	1.00000	0.99593	1.00000
69	1.00000	0.49984	1.00000	0.99329	0.99993	0.99993	0.99153	1.00000	0.92050	1.00000
	1.00000	0.49984	1.00000	0.99312	1.00000	0.99993	0.99993	1.00000	0.99625	1.00000
70	1.00000	0.49484	1.00000	0.99319	0.99994	0.99993	0.99211	1.00000	0.92350	1.00000
	1.00000	0.49484	1.00000	0.99302	1.00000	0.99993	0.99994	1.00000	0.99654	1.00000
71	1.00000	0.48989	1.00000	0.99309	0.99995	0.99993	0.99266	1.00000	0.92638	1.00000
	1.00000	0.48989	1.00000	0.99292	1.00000	0.99993	0.99995	1.00000	0.99681	1.00000
72	1.00000	0.48499	1.00000	0.99299	0.99995	0.99993	0.99317	1.00000	0.92916	1.00000
	1.00000	0.48499	1.00000	0.99283	1.00000	0.99993	0.99995	1.00000	0.99706	1.00000
73	1.00000	0.48014	1.00000	0.99289	0.99996	0.99993	0.99364	1.00000	0.93183	1.00000
	1.00000	0.48014	1.00000	0.99273	1.00000	0.99993	0.99996	1.00000	0.99729	1.00000
74	1.00000	0.47534	1.00000	0.99280	0.99997	0.99993	0.99408	1.00000	0.93440	1.00000
	1.00000	0.47534	1.00000	0.99263	1.00000	0.99993	0.99997	1.00000	0.99750	1.00000
75	1.00000	0.47059	1.00000	0.99270	0.99997	0.99993	0.99448	1.00000	0.93687	1.00000
	1.00000	0.47059	1.00000	0.99253	1.00000	0.99993	0.99997	1.00000	0.99769	1.00000
76	1.00000	0.46588	1.00000	0.99260	0.99997	0.99993	0.99487	1.00000	0.93925	1.00000
	1.00000	0.46588	1.00000	0.99243	1.00000	0.99992	0.99997	1.00000	0.99787	1.00000
77	1.00000	0.46122	1.00000	0.99250	0.99998	0.99993	0.99522	1.00000	0.94154	1.00000
	1.00000	0.46122	1.00000	0.99233	1.00000	0.99992	0.99998	1.00000	0.99804	1.00000
78	1.00000	0.45661	1.00000	0.99240	0.99998	0.99992	0.99555	1.00000	0.94374	1.00000
	1.00000	0.45661	1.00000	0.99223	1.00000	0.99992	0.99998	1.00000	0.99819	1.00000
79	1.00000	0.45204	1.00000	0.99230	0.99998	0.99992	0.99586	1.00000	0.94586	1.00000
	1.00000	0.45204	1.00000	0.99213	1.00000	0.99992	0.99998	1.00000	0.99833	1.00000
80	1.00000	0.44752	1.00000	0.99221	0.99999	0.99992	0.99614	1.00000	0.94790	1.00000
	1.00000	0.44752	1.00000	0.99203	1.00000	0.99992	0.99998	1.00000	0.99846	1.00000
81	1.00000	0.44305	1.00000	0.99211	0.99999	0.99992	0.99641	1.00000	0.94987	1.00000
	1.00000	0.44305	1.00000	0.99193	1.00000	0.99992	0.99999	1.00000	0.99858	1.00000
82	1.00000	0.43862	1.00000	0.99201	0.99999	0.99992	0.99666	1.00000	0.95176	1.00000
	1.00000	0.43862	1.00000	0.99183	1.00000	0.99992	0.99999	1.00000	0.99869	1.00000
83	1.00000	0.43423	1.00000	0.99191	0.99999	0.99992	0.99689	1.00000	0.95357	1.00000
	1.00000	0.43423	1.00000	0.99173	1.00000	0.99992	0.99999	1.00000	0.99879	1.00000
84	1.00000	0.42989	1.00000	0.99181	0.99999	0.99992	0.99710	1.00000	0.95532	1.00000
	1.00000	0.42989	1.00000	0.99163	1.00000	0.99992	0.99999	1.00000	0.99889	1.00000
85	1.00000	0.42559	1.00000	0.99172	0.99999	0.99992	0.99730	1.00000	0.95701	1.00000
	1.00000	0.42559	1.00000	0.99154	1.00000	0.99992	0.99999	1.00000	0.99897	1.00000
86	1.00000	0.42133	1.00000	0.99162	0.99999	0.99992	0.99749	1.00000	0.95863	1.00000
	1.00000	0.42133	1.00000	0.99144	1.00000	0.99991	0.99999	1.00000	0.99905	1.00000
87	1.00000	0.41712	1.00000	0.99152	0.99999	0.99992	0.99766	1.00000	0.96019	1.00000
	1.00000	0.41712	1.00000	0.99134	1.00000	0.99991	0.99999	1.00000	0.99913	1.00000
88	1.00000	0.41295	1.00000	0.99142	1.00000	0.99991	0.99782	1.00000	0.96169	1.00000
	1.00000	0.41295	1.00000	0.99124	1.00000	0.99991	1.00000	1.00000	0.99919	1.00000
89	1.00000	0.40882	1.00000	0.99132	1.00000	0.99991	0.99797	1.00000	0.96313	1.00000
	1.00000	0.40882	1.00000	0.99114	1.00000	0.99991	1.00000	1.00000	0.99926	1.00000
90	1.00000	0.40473	1.00000	0.99122	1.00000	0.99991	0.99811	1.00000	0.96452	1.00000
	1.00000	0.40473	1.00000	0.99104	1.00000	0.99991	1.00000	1.00000	0.99931	1.00000

91	1.00000	0.40068	1.00000	0.99113	1.00000	0.99991	0.99824	1.00000	0.96586 1.00000
	1.00000	0.40068	1.00000	0.99094	1.00000	0.99991	1.00000	1.00000	0.99937 1.00000
92	1.00000	0.39668	1.00000	0.99103	1.00000	0.99991	0.99836	1.00000	0.96715 1.00000
	1.00000	0.39668	1.00000	0.99084	1.00000	0.99991	1.00000	1.00000	0.99942 1.00000
93	1.00000	0.39271	1.00000	0.99093	1.00000	0.99991	0.99848	1.00000	0.96838 1.00000
	1.00000	0.39271	1.00000	0.99074	1.00000	0.99991	1.00000	1.00000	0.99946 1.00000
94	1.00000	0.38878	1.00000	0.99083	1.00000	0.99991	0.99858	1.00000	0.96958 1.00000
	1.00000	0.38878	1.00000	0.99064	1.00000	0.99991	1.00000	1.00000	0.99950 1.00000
95	1.00000	0.38490	1.00000	0.99073	1.00000	0.99991	0.99868	1.00000	0.97072 1.00000
	1.00000	0.38490	1.00000	0.99054	1.00000	0.99991	1.00000	1.00000	0.99954 1.00000
96	1.00000	0.38105	1.00000	0.99064	1.00000	0.99991	0.99877	1.00000	0.97183 1.00000
	1.00000	0.38105	1.00000	0.99045	1.00000	0.99990	1.00000	1.00000	0.99958 1.00000
97	1.00000	0.37724	1.00000	0.99054	1.00000	0.99991	0.99886	1.00000	0.97289 1.00000
	1.00000	0.37724	1.00000	0.99035	1.00000	0.99990	1.00000	1.00000	0.99961 1.00000
98	1.00000	0.37346	1.00000	0.99044	1.00000	0.99990	0.99893	1.00000	0.97391 1.00000
	1.00000	0.37346	1.00000	0.99025	1.00000	0.99990	1.00000	1.00000	0.99964 1.00000
99	1.00000	0.36973	1.00000	0.99034	1.00000	0.99990	0.99901	1.00000	0.97489 1.00000
	1.00000	0.36973	1.00000	0.99015	1.00000	0.99990	1.00000	1.00000	0.99967 1.00000
100	1.00000	0.36603	1.00000	0.99024	1.00000	0.99990	0.99908	1.00000	0.97584 1.00000
	1.00000	0.36603	1.00000	0.99005	1.00000	0.99990	1.00000	1.00000	0.99969 1.00000

********** p a r a m e t e r s **********

value for r : 0.990000
value for s : 0.400000
value for m_max : 1000
value for k_max : 5
****

The resulting table contains in every field four values

s_net r_net for security/reliability of the MIX-network
s_com r_com as above but of a series-parallel circuit

the horizontal index is for values of 'k'; the vertical index is for values of 'm'

	1		2		3		4		5	
1	0.40000	0.99000	0.00000	1.00000	0.00000	1.00000	0.00000	1.00000	0.00000	1.00000
	0.40000	0.99000	0.00000	1.00000	0.00000	1.00000	0.00000	1.00000	0.00000	1.00000
2	0.64000	0.98010	0.16000	0.99990	0.00000	1.00000	0.00000	1.00000	0.00000	1.00000
	0.64000	0.98010	0.29440	0.99980	0.00000	1.00000	0.00000	1.00000	0.00000	1.00000
3	0.78400	0.97030	0.25600	0.99980	0.06400	1.00000	0.00000	1.00000	0.00000	1.00000
	0.78400	0.97030	0.40730	0.99970	0.17997	1.00000	0.00000	1.00000	0.00000	1.00000
4	0.87040	0.96060	0.35200	0.99970	0.10240	1.00000	0.02560	1.00000	0.00000	1.00000
	0.87040	0.96060	0.50213	0.99960	0.23246	1.00000	0.09853	1.00000	0.00000	1.00000
5	0.92224	0.95099	0.43264	0.99960	0.14080	1.00000	0.04096	1.00000	0.01024	1.00000
	0.92224	0.95099	0.58179	0.99950	0.28158	1.00000	0.12161	1.00000	0.05016	1.00000
6	0.95334	0.94148	0.50406	0.99950	0.17920	1.00000	0.05632	1.00000	0.01638	1.00000
	0.95334	0.94148	0.64870	0.99940	0.32756	0.99999	0.14410	1.00000	0.05989	1.00000
7	0.97201	0.93207	0.56627	0.99941	0.21514	1.00000	0.07168	1.00000	0.02253	1.00000
	0.97201	0.93207	0.70491	0.99930	0.37059	0.99999	0.16601	1.00000	0.06952	1.00000
8	0.98320	0.92274	0.62074	0.99931	0.24961	0.99999	0.08704	1.00000	0.02867	1.00000
	0.98320	0.92274	0.75212	0.99920	0.41088	0.99999	0.18736	1.00000	0.07904	1.00000
9	0.98992	0.91352	0.66835	0.99921	0.28260	0.99999	0.10201	1.00000	0.03482	1.00000
	0.98992	0.91352	0.79178	0.99910	0.44858	0.99999	0.20816	1.00000	0.08847	1.00000
10	0.99395	0.90438	0.70999	0.99911	0.31412	0.99999	0.11674	1.00000	0.04096	1.00000
	0.99395	0.90438	0.82510	0.99900	0.48387	0.99999	0.22843	1.00000	0.09781	1.00000
11	0.99637	0.89534	0.74640	0.99901	0.34426	0.99999	0.13123	1.00000	0.04704	1.00000
	0.99637	0.89534	0.85308	0.99890	0.51690	0.99999	0.24819	1.00000	0.10705	1.00000
12	0.99782	0.88638	0.77823	0.99891	0.37308	0.99999	0.14549	1.00000	0.05308	1.00000
	0.99782	0.88638	0.87659	0.99880	0.54782	0.99999	0.26743	1.00000	0.11619	1.00000
13	0.99869	0.87752	0.80608	0.99881	0.40062	0.99999	0.15951	1.00000	0.05909	1.00000
	0.99869	0.87752	0.89634	0.99870	0.57676	0.99999	0.28619	1.00000	0.12524	1.00000
14	0.99922	0.86875	0.83042	0.99871	0.42696	0.99999	0.17331	1.00000	0.06506	1.00000
	0.99922	0.86875	0.91292	0.99860	0.60385	0.99999	0.30446	1.00000	0.13420	1.00000
15	0.99953	0.86006	0.85171	0.99861	0.45214	0.99999	0.18687	1.00000	0.07099	1.00000
	0.99953	0.86006	0.92685	0.99850	0.62920	0.99999	0.32227	1.00000	0.14306	1.00000

16	0.99972 0.85146	0.87033 0.99851	0.47622 0.99999	0.20022 1.00000	0.07688 1.00000
	0.99972 0.85146	0.93856 0.99840	0.65293 0.99998	0.33962 1.00000	0.15184 1.00000
17	0.99983 0.84294	0.88661 0.99842	0.49923 0.99999	0.21334 1.00000	0.08274 1.00000
	0.99983 0.84294	0.94839 0.99830	0.67515 0.99998	0.35652 1.00000	0.16052 1.00000
18	0.99990 0.83451	0.90084 0.99832	0.52124 0.99998	0.22625 1.00000	0.08855 1.00000
	0.99990 0.83451	0.95665 0.99820	0.69594 0.99998	0.37300 1.00000	0.16912 1.00000
19	0.99994 0.82617	0.91329 0.99822	0.54227 0.99998	0.23895 1.00000	0.09433 1.00000
	0.99994 0.82617	0.96358 0.99810	0.71540 0.99998	0.38905 1.00000	0.17763 1.00000
20	0.99996 0.81791	0.92418 0.99812	0.56239 0.99998	0.25144 1.00000	0.10008 1.00000
	0.99996 0.81791	0.96941 0.99800	0.73361 0.99998	0.40469 1.00000	0.18605 1.00000
21	0.99998 0.80973	0.93370 0.99802	0.58162 0.99998	0.26373 1.00000	0.10579 1.00000
	0.99998 0.80973	0.97430 0.99790	0.75066 0.99998	0.41993 1.00000	0.19438 1.00000
22	0.99999 0.80163	0.94202 0.99792	0.60000 0.99998	0.27581 1.00000	0.11146 1.00000
	0.99999 0.80163	0.97842 0.99780	0.76662 0.99998	0.43478 1.00000	0.20263 1.00000
23	0.99999 0.79361	0.94930 0.99782	0.61758 0.99998	0.28769 1.00000	0.11709 1.00000
	0.99999 0.79361	0.98187 0.99770	0.78155 0.99998	0.44925 1.00000	0.21080 1.00000
24	1.00000 0.78568	0.95566 0.99772	0.63438 0.99998	0.29938 1.00000	0.12269 1.00000
	1.00000 0.78568	0.98477 0.99760	0.79553 0.99998	0.46335 1.00000	0.21888 1.00000
25	1.00000 0.77782	0.96123 0.99763	0.65045 0.99998	0.31088 1.00000	0.12826 1.00000
	1.00000 0.77782	0.98721 0.99750	0.80862 0.99998	0.47708 1.00000	0.22688 1.00000
26	1.00000 0.77004	0.96610 0.99753	0.66581 0.99998	0.32219 1.00000	0.13379 1.00000
	1.00000 0.77004	0.98925 0.99740	0.82087 0.99997	0.49047 1.00000	0.23480 1.00000
27	1.00000 0.76234	0.97035 0.99743	0.68049 0.99998	0.33331 1.00000	0.13928 1.00000
	1.00000 0.76234	0.99097 0.99730	0.83233 0.99997	0.50351 1.00000	0.24263 1.00000
28	1.00000 0.75472	0.97408 0.99733	0.69453 0.99997	0.34426 1.00000	0.14474 1.00000
	1.00000 0.75472	0.99242 0.99720	0.84306 0.99997	0.51622 1.00000	0.25039 1.00000
29	1.00000 0.74717	0.97733 0.99723	0.70796 0.99997	0.35502 1.00000	0.15016 1.00000
	1.00000 0.74717	0.99363 0.99710	0.85311 0.99997	0.52861 1.00000	0.25806 1.00000
30	1.00000 0.73970	0.98018 0.99713	0.72079 0.99997	0.36560 1.00000	0.15555 1.00000
	1.00000 0.73970	0.99465 0.99700	0.86251 0.99997	0.54068 1.00000	0.26566 1.00000
31	1.00000 0.73230	0.98267 0.99703	0.73306 0.99997	0.37601 1.00000	0.16091 1.00000
	1.00000 0.73230	0.99551 0.99690	0.87131 0.99997	0.55244 1.00000	0.27318 1.00000
32	1.00000 0.72498	0.98484 0.99693	0.74479 0.99997	0.38625 1.00000	0.16623 1.00000
	1.00000 0.72498	0.99622 0.99680	0.87954 0.99997	0.56389 1.00000	0.28062 1.00000
33	1.00000 0.71773	0.98674 0.99684	0.75600 0.99997	0.39633 1.00000	0.17152 1.00000
	1.00000 0.71773	0.99683 0.99671	0.88725 0.99997	0.57506 1.00000	0.28799 1.00000
34	1.00000 0.71055	0.98841 0.99674	0.76672 0.99997	0.40623 1.00000	0.17678 1.00000
	1.00000 0.71055	0.99734 0.99661	0.89447 0.99997	0.58594 1.00000	0.29528 1.00000
35	1.00000 0.70345	0.98986 0.99664	0.77697 0.99997	0.41598 1.00000	0.18200 1.00000
	1.00000 0.70345	0.99776 0.99651	0.90122 0.99997	0.59654 1.00000	0.30250 1.00000
36	1.00000 0.69641	0.99114 0.99654	0.78677 0.99997	0.42556 1.00000	0.18719 1.00000
	1.00000 0.69641	0.99812 0.99641	0.90754 0.99996	0.60686 1.00000	0.30964 1.00000
37	1.00000 0.68945	0.99225 0.99644	0.79614 0.99997	0.43499 1.00000	0.19234 1.00000
	1.00000 0.68945	0.99842 0.99631	0.91346 0.99996	0.61693 1.00000	0.31671 1.00000
38	1.00000 0.68255	0.99322 0.99634	0.80510 0.99996	0.44426 1.00000	0.19746 1.00000
	1.00000 0.68255	0.99867 0.99621	0.91900 0.99996	0.62674 1.00000	0.32371 1.00000
39	1.00000 0.67573	0.99407 0.99624	0.81367 0.99996	0.45338 1.00000	0.20255 1.00000
	1.00000 0.67573	0.99889 0.99611	0.92418 0.99996	0.63629 1.00000	0.33063 1.00000
40	1.00000 0.66897	0.99482 0.99614	0.82185 0.99996	0.46235 1.00000	0.20761 1.00000
	1.00000 0.66897	0.99906 0.99601	0.92904 0.99996	0.64560 1.00000	0.33748 1.00000
41	1.00000 0.66228	0.99547 0.99605	0.82968 0.99996	0.47117 1.00000	0.21264 1.00000
	1.00000 0.66228	0.99921 0.99591	0.93358 0.99996	0.65467 1.00000	0.34427 1.00000
42	1.00000 0.65566	0.99604 0.99595	0.83717 0.99996	0.47985 1.00000	0.21763 1.00000
	1.00000 0.65566	0.99934 0.99581	0.93783 0.99996	0.66352 1.00000	0.35098 1.00000
43	1.00000 0.64910	0.99653 0.99585	0.84432 0.99996	0.48839 1.00000	0.22259 1.00000
	1.00000 0.64910	0.99945 0.99571	0.94181 0.99996	0.67213 1.00000	0.35763 1.00000
44	1.00000 0.64261	0.99697 0.99575	0.85116 0.99996	0.49679 1.00000	0.22752 1.00000
	1.00000 0.64261	0.99953 0.99561	0.94553 0.99996	0.68052 1.00000	0.36421 1.00000
45	1.00000 0.63619	0.99735 0.99565	0.85770 0.99996	0.50504 1.00000	0.23242 1.00000
	1.00000 0.63619	0.99961 0.99551	0.94902 0.99996	0.68870 1.00000	0.37072 1.00000
46	1.00000 0.62982	0.99768 0.99555	0.86396 0.99996	0.51317 1.00000	0.23729 1.00000
	1.00000 0.62982	0.99967 0.99541	0.95228 0.99995	0.69667 1.00000	0.37716 1.00000
47	1.00000 0.62353	0.99797 0.99545	0.86993 0.99996	0.52116 1.00000	0.24213 1.00000
	1.00000 0.62353	0.99972 0.99531	0.95534 0.99995	0.70444 1.00000	0.38354 1.00000

48	1.00000 0.61729	0.99823 0.99536	0.87565 0.99995	0.52901 1.00000	0.24694 1.00000
	1.00000 0.61729	0.99977 0.99521	0.95819 0.99995	0.71200 1.00000	0.38985 1.00000
49	1.00000 0.61112	0.99845 0.99526	0.88111 0.99995	0.53674 1.00000	0.25171 1.00000
	1.00000 0.61112	0.99981 0.99511	0.96087 0.99995	0.71937 1.00000	0.39610 1.00000
50	1.00000 0.60501	0.99864 0.99516	0.88634 0.99995	0.54435 1.00000	0.25646 1.00000
	1.00000 0.60501	0.99984 0.99501	0.96337 0.99995	0.72656 1.00000	0.40228 1.00000
51	1.00000 0.59896	0.99882 0.99506	0.89133 0.99995	0.55182 1.00000	0.26118 1.00000
	1.00000 0.59896	0.99986 0.99491	0.96572 0.99995	0.73356 1.00000	0.40840 1.00000
52	1.00000 0.59297	0.99896 0.99496	0.89611 0.99995	0.55918 1.00000	0.26586 1.00000
	1.00000 0.59297	0.99988 0.99481	0.96791 0.99995	0.74038 1.00000	0.41446 1.00000
53	1.00000 0.58704	0.99909 0.99486	0.90067 0.99995	0.56641 1.00000	0.27052 1.00000
	1.00000 0.58704	0.99990 0.99471	0.96997 0.99995	0.74703 1.00000	0.42046 1.00000
54	1.00000 0.58117	0.99921 0.99476	0.90504 0.99995	0.57353 1.00000	0.27514 1.00000
	1.00000 0.58117	0.99992 0.99461	0.97189 0.99995	0.75350 1.00000	0.42639 1.00000
55	1.00000 0.57535	0.99931 0.99467	0.90921 0.99995	0.58053 1.00000	0.27974 1.00000
	1.00000 0.57535	0.99993 0.99451	0.97369 0.99995	0.75981 1.00000	0.43227 1.00000
56	1.00000 0.56960	0.99939 0.99457	0.91320 0.99995	0.58741 1.00000	0.28431 1.00000
	1.00000 0.56960	0.99994 0.99442	0.97537 0.99994	0.76596 1.00000	0.43808 1.00000
57	1.00000 0.56391	0.99947 0.99447	0.91701 0.99995	0.59418 1.00000	0.28885 1.00000
	1.00000 0.56391	0.99995 0.99432	0.97695 0.99994	0.77195 1.00000	0.44383 1.00000
58	1.00000 0.55827	0.99954 0.99437	0.92066 0.99994	0.60084 1.00000	0.29336 1.00000
	1.00000 0.55827	0.99996 0.99422	0.97842 0.99994	0.77779 1.00000	0.44953 1.00000
59	1.00000 0.55268	0.99959 0.99427	0.92415 0.99994	0.60739 1.00000	0.29784 1.00000
	1.00000 0.55268	0.99997 0.99412	0.97980 0.99994	0.78348 1.00000	0.45517 1.00000
60	1.00000 0.54716	0.99965 0.99417	0.92748 0.99994	0.61384 1.00000	0.30230 1.00000
	1.00000 0.54716	0.99997 0.99402	0.98110 0.99994	0.78902 1.00000	0.46075 1.00000
61	1.00000 0.54169	0.99969 0.99408	0.93067 0.99994	0.62017 1.00000	0.30672 1.00000
	1.00000 0.54169	0.99998 0.99392	0.98231 0.99994	0.79442 1.00000	0.46627 1.00000
62	1.00000 0.53627	0.99973 0.99398	0.93371 0.99994	0.62641 1.00000	0.31112 1.00000
	1.00000 0.53627	0.99998 0.99382	0.98344 0.99994	0.79969 1.00000	0.47173 1.00000
63	1.00000 0.53091	0.99976 0.99388	0.93663 0.99994	0.63254 1.00000	0.31549 1.00000
	1.00000 0.53091	0.99998 0.99372	0.98450 0.99994	0.80481 1.00000	0.47714 1.00000
64	1.00000 0.52560	0.99979 0.99378	0.93941 0.99994	0.63857 1.00000	0.31983 1.00000
	1.00000 0.52560	0.99999 0.99362	0.98549 0.99994	0.80981 1.00000	0.48250 1.00000
65	1.00000 0.52034	0.99982 0.99368	0.94207 0.99994	0.64450 1.00000	0.32414 1.00000
	1.00000 0.52034	0.99999 0.99352	0.98642 0.99994	0.81468 1.00000	0.48780 1.00000
66	1.00000 0.51514	0.99984 0.99358	0.94462 0.99994	0.65033 1.00000	0.32843 1.00000
	1.00000 0.51514	0.99999 0.99342	0.98729 0.99993	0.81942 1.00000	0.49304 1.00000
67	1.00000 0.50999	0.99986 0.99348	0.94705 0.99994	0.65607 1.00000	0.33269 1.00000
	1.00000 0.50999	0.99999 0.99332	0.98810 0.99993	0.82405 1.00000	0.49823 1.00000
68	1.00000 0.50489	0.99988 0.99339	0.94938 0.99993	0.66172 1.00000	0.33692 1.00000
	1.00000 0.50489	0.99999 0.99322	0.98886 0.99993	0.82855 1.00000	0.50337 1.00000
69	1.00000 0.49984	0.99989 0.99329	0.95160 0.99993	0.66727 1.00000	0.34113 1.00000
	1.00000 0.49984	0.99999 0.99312	0.98958 0.99993	0.83294 1.00000	0.50846 1.00000
70	1.00000 0.49484	0.99991 0.99319	0.95373 0.99993	0.67273 1.00000	0.34531 1.00000
	1.00000 0.49484	0.99999 0.99302	0.99024 0.99993	0.83722 1.00000	0.51349 1.00000
71	1.00000 0.48989	0.99992 0.99309	0.95576 0.99993	0.67810 1.00000	0.34946 1.00000
	1.00000 0.48989	1.00000 0.99292	0.99087 0.99993	0.84138 1.00000	0.51847 1.00000
72	1.00000 0.48499	0.99993 0.99299	0.95771 0.99993	0.68338 1.00000	0.35359 1.00000
	1.00000 0.48499	1.00000 0.99283	0.99145 0.99993	0.84544 1.00000	0.52340 1.00000
73	1.00000 0.48014	0.99994 0.99289	0.95956 0.99993	0.68858 1.00000	0.35768 1.00000
	1.00000 0.48014	1.00000 0.99273	0.99200 0.99993	0.84940 1.00000	0.52828 1.00000
74	1.00000 0.47534	0.99995 0.99280	0.96134 0.99993	0.69369 1.00000	0.36176 1.00000
	1.00000 0.47534	1.00000 0.99263	0.99251 0.99993	0.85326 1.00000	0.53311 1.00000
75	1.00000 0.47059	0.99995 0.99270	0.96304 0.99993	0.69872 1.00000	0.36581 1.00000
	1.00000 0.47059	1.00000 0.99253	0.99299 0.99993	0.85701 1.00000	0.53789 1.00000
76	1.00000 0.46588	0.99996 0.99260	0.96466 0.99993	0.70366 1.00000	0.36983 1.00000
	1.00000 0.46588	1.00000 0.99243	0.99344 0.99992	0.86067 1.00000	0.54263 1.00000
77	1.00000 0.46122	0.99996 0.99250	0.96622 0.99993	0.70852 1.00000	0.37383 1.00000
	1.00000 0.46122	1.00000 0.99233	0.99386 0.99992	0.86424 1.00000	0.54731 1.00000
78	1.00000 0.45661	0.99997 0.99240	0.96770 0.99992	0.71331 1.00000	0.37780 1.00000
	1.00000 0.45661	1.00000 0.99223	0.99425 0.99992	0.86772 1.00000	0.55194 1.00000
79	1.00000 0.45204	0.99997 0.99230	0.96912 0.99992	0.71801 1.00000	0.38174 1.00000
	1.00000 0.45204	1.00000 0.99213	0.99462 0.99992	0.87110 1.00000	0.55653 1.00000

80	1.00000	0.44752	0.99998	0.99221	0.97048	0.99992	0.72264	1.00000	0.38567	1.00000
	1.00000	0.44752	1.00000	0.99203	0.99496	0.99992	0.87440	1.00000	0.56107	1.00000
81	1.00000	0.44305	0.99998	0.99211	0.97178	0.99992	0.72719	1.00000	0.38956	1.00000
	1.00000	0.44305	1.00000	0.99193	0.99529	0.99992	0.87762	1.00000	0.56557	1.00000
82	1.00000	0.43862	0.99998	0.99201	0.97302	0.99992	0.73167	1.00000	0.39343	1.00000
	1.00000	0.43862	1.00000	0.99183	0.99559	0.99992	0.88075	1.00000	0.57002	1.00000
83	1.00000	0.43423	0.99998	0.99191	0.97420	0.99992	0.73607	1.00000	0.39728	1.00000
	1.00000	0.43423	1.00000	0.99173	0.99587	0.99992	0.88380	1.00000	0.57442	1.00000
84	1.00000	0.42989	0.99999	0.99181	0.97533	0.99992	0.74040	1.00000	0.40110	1.00000
	1.00000	0.42989	1.00000	0.99163	0.99613	0.99992	0.88678	1.00000	0.57878	1.00000
85	1.00000	0.42559	0.99999	0.99172	0.97642	0.99992	0.74466	1.00000	0.40490	1.00000
	1.00000	0.42559	1.00000	0.99154	0.99638	0.99992	0.88968	1.00000	0.58309	1.00000
86	1.00000	0.42133	0.99999	0.99162	0.97745	0.99992	0.74885	1.00000	0.40868	1.00000
	1.00000	0.42133	1.00000	0.99144	0.99661	0.99991	0.89250	1.00000	0.58736	1.00000
87	1.00000	0.41712	0.99999	0.99152	0.97845	0.99992	0.75298	1.00000	0.41243	1.00000
	1.00000	0.41712	1.00000	0.99134	0.99683	0.99991	0.89525	1.00000	0.59159	1.00000
88	1.00000	0.41295	0.99999	0.99142	0.97939	0.99991	0.75703	1.00000	0.41615	1.00000
	1.00000	0.41295	1.00000	0.99124	0.99703	0.99991	0.89793	1.00000	0.59577	1.00000
89	1.00000	0.40882	0.99999	0.99132	0.98030	0.99991	0.76102	1.00000	0.41986	1.00000
	1.00000	0.40882	1.00000	0.99114	0.99722	0.99991	0.90055	1.00000	0.59991	1.00000
90	1.00000	0.40473	0.99999	0.99122	0.98116	0.99991	0.76494	1.00000	0.42354	1.00000
	1.00000	0.40473	1.00000	0.99104	0.99740	0.99991	0.90309	1.00000	0.60400	1.00000
91	1.00000	0.40068	0.99999	0.99113	0.98199	0.99991	0.76880	1.00000	0.42719	1.00000
	1.00000	0.40068	1.00000	0.99094	0.99757	0.99991	0.90557	1.00000	0.60806	1.00000
92	1.00000	0.39668	1.00000	0.99103	0.98278	0.99991	0.77259	1.00000	0.43083	1.00000
	1.00000	0.39668	1.00000	0.99084	0.99772	0.99991	0.90799	1.00000	0.61207	1.00000
93	1.00000	0.39271	1.00000	0.99093	0.98354	0.99991	0.77632	1.00000	0.43444	1.00000
	1.00000	0.39271	1.00000	0.99074	0.99787	0.99991	0.91035	1.00000	0.61604	1.00000
94	1.00000	0.38878	1.00000	0.99083	0.98426	0.99991	0.77999	1.00000	0.43802	1.00000
	1.00000	0.38878	1.00000	0.99064	0.99801	0.99991	0.91264	1.00000	0.61998	1.00000
95	1.00000	0.38490	1.00000	0.99073	0.98495	0.99991	0.78360	1.00000	0.44159	1.00000
	1.00000	0.38490	1.00000	0.99054	0.99813	0.99991	0.91488	1.00000	0.62387	1.00000
96	1.00000	0.38105	1.00000	0.99064	0.98562	0.99991	0.78716	1.00000	0.44513	1.00000
	1.00000	0.38105	1.00000	0.99045	0.99825	0.99990	0.91706	1.00000	0.62772	1.00000
97	1.00000	0.37724	1.00000	0.99054	0.98625	0.99991	0.79065	1.00000	0.44865	1.00000
	1.00000	0.37724	1.00000	0.99035	0.99836	0.99990	0.91918	1.00000	0.63153	1.00000
98	1.00000	0.37346	1.00000	0.99044	0.98685	0.99990	0.79408	1.00000	0.45215	1.00000
	1.00000	0.37346	1.00000	0.99025	0.99847	0.99990	0.92125	1.00000	0.63530	1.00000
99	1.00000	0.36973	1.00000	0.99034	0.98743	0.99990	0.79746	1.00000	0.45562	1.00000
	1.00000	0.36973	1.00000	0.99015	0.99857	0.99990	0.92327	1.00000	0.63904	1.00000
100	1.00000	0.36603	1.00000	0.99024	0.98798	0.99990	0.80079	1.00000	0.45907	1.00000
	1.00000	0.36603	1.00000	0.99005	0.99866	0.99990	0.92523	1.00000	0.64274	1.00000
110	1.00000	0.33103	1.00000	0.98926	0.99233	0.99989	0.83117	1.00000	0.49242	1.00000
	1.00000	0.33103	1.00000	0.98906	0.99931	0.99989	0.94231	1.00000	0.67768	1.00000
120	1.00000	0.29938	1.00000	0.98828	0.99511	0.99988	0.85692	1.00000	0.52371	1.00000
	1.00000	0.29938	1.00000	0.98807	0.99964	0.99988	0.95549	1.00000	0.70920	1.00000
130	1.00000	0.27075	1.00000	0.98731	0.99688	0.99987	0.87874	1.00000	0.55307	1.00000
	1.00000	0.27075	1.00000	0.98708	0.99982	0.99987	0.96566	1.00000	0.73765	1.00000
140	1.00000	0.24487	1.00000	0.98633	0.99801	0.99986	0.89723	1.00000	0.58062	1.00000
	1.00000	0.24487	1.00000	0.98610	0.99990	0.99986	0.97350	1.00000	0.76331	1.00000
150	1.00000	0.22145	1.00000	0.98535	0.99873	0.99985	0.91290	1.00000	0.60648	1.00000
	1.00000	0.22145	1.00000	0.98511	0.99995	0.99985	0.97955	1.00000	0.78646	1.00000
160	1.00000	0.20028	1.00000	0.98438	0.99919	0.99984	0.92619	1.00000	0.63073	1.00000
	1.00000	0.20028	1.00000	0.98413	0.99997	0.99984	0.98423	1.00000	0.80734	1.00000
170	1.00000	0.18113	1.00000	0.98340	0.99948	0.99983	0.93744	1.00000	0.65350	1.00000
	1.00000	0.18113	1.00000	0.98314	0.99999	0.99983	0.98783	1.00000	0.82619	1.00000
180	1.00000	0.16381	1.00000	0.98243	0.99967	0.99982	0.94698	1.00000	0.67486	1.00000
	1.00000	0.16381	1.00000	0.98216	0.99999	0.99982	0.99061	1.00000	0.84319	1.00000
190	1.00000	0.14814	1.00000	0.98146	0.99979	0.99981	0.95507	1.00000	0.69490	1.00000
	1.00000	0.14814	1.00000	0.98118	1.00000	0.99981	0.99275	1.00000	0.85852	1.00000
200	1.00000	0.13398	1.00000	0.98049	0.99987	0.99980	0.96192	1.00000	0.71371	1.00000
	1.00000	0.13398	1.00000	0.98020	1.00000	0.99980	0.99441	1.00000	0.87236	1.00000
210	1.00000	0.12117	1.00000	0.97952	0.99991	0.99979	0.96773	1.00000	0.73136	1.00000
	1.00000	0.12117	1.00000	0.97922	1.00000	0.99979	0.99569	1.00000	0.88485	1.00000

220	1.00000 0.10958	1.00000 0.97855	0.99995 0.99978	0.97265 1.00000	0.74792 1.00000
	1.00000 0.10958	1.00000 0.97824	1.00000 0.99978	0.99667 1.00000	0.89611 1.00000
230	1.00000 0.09910	1.00000 0.97758	0.99997 0.99977	0.97682 1.00000	0.76346 1.00000
	1.00000 0.09910	1.00000 0.97726	1.00000 0.99977	0.99743 1.00000	0.90627 1.00000
240	1.00000 0.08963	1.00000 0.97661	0.99998 0.99976	0.98036 1.00000	0.77804 1.00000
	1.00000 0.08963	1.00000 0.97628	1.00000 0.99976	0.99802 1.00000	0.91544 1.00000
250	1.00000 0.08106	1.00000 0.97564	0.99999 0.99975	0.98335 1.00000	0.79172 1.00000
	1.00000 0.08106	1.00000 0.97531	1.00000 0.99975	0.99847 1.00000	0.92371 1.00000
260	1.00000 0.07331	1.00000 0.97468	0.99999 0.99974	0.98589 1.00000	0.80456 1.00000
	1.00000 0.07331	1.00000 0.97433	1.00000 0.99974	0.99882 1.00000	0.93117 1.00000
270	1.00000 0.06630	1.00000 0.97371	0.99999 0.99973	0.98804 1.00000	0.81661 1.00000
	1.00000 0.06630	1.00000 0.97336	1.00000 0.99973	0.99909 1.00000	0.93790 1.00000
280	1.00000 0.05996	1.00000 0.97275	1.00000 0.99972	0.98987 1.00000	0.82792 1.00000
	1.00000 0.05996	1.00000 0.97239	1.00000 0.99972	0.99930 1.00000	0.94398 1.00000
290	1.00000 0.05423	1.00000 0.97179	1.00000 0.99971	0.99141 1.00000	0.83853 1.00000
	1.00000 0.05423	1.00000 0.97142	1.00000 0.99971	0.99946 1.00000	0.94946 1.00000
300	1.00000 0.04904	1.00000 0.97082	1.00000 0.99971	0.99272 1.00000	0.84848 1.00000
	1.00000 0.04904	1.00000 0.97044	1.00000 0.99970	0.99958 1.00000	0.95440 1.00000
310	1.00000 0.04435	1.00000 0.96986	1.00000 0.99970	0.99383 1.00000	0.85782 1.00000
	1.00000 0.04435	1.00000 0.96947	1.00000 0.99969	0.99968 1.00000	0.95886 1.00000
320	1.00000 0.04011	1.00000 0.96890	1.00000 0.99969	0.99477 1.00000	0.86658 1.00000
	1.00000 0.04011	1.00000 0.96851	1.00000 0.99968	0.99975 1.00000	0.96288 1.00000
330	1.00000 0.03628	1.00000 0.96794	1.00000 0.99968	0.99557 1.00000	0.87481 1.00000
	1.00000 0.03628	1.00000 0.96754	1.00000 0.99967	0.99981 1.00000	0.96651 1.00000
340	1.00000 0.03281	1.00000 0.96699	1.00000 0.99967	0.99625 1.00000	0.88253 1.00000
	1.00000 0.03281	1.00000 0.96657	1.00000 0.99966	0.99985 1.00000	0.96979 1.00000
350	1.00000 0.02967	1.00000 0.96603	1.00000 0.99966	0.99682 1.00000	0.88977 1.00000
	1.00000 0.02967	1.00000 0.96560	1.00000 0.99965	0.99989 1.00000	0.97274 1.00000
360	1.00000 0.02683	1.00000 0.96507	1.00000 0.99965	0.99730 1.00000	0.89656 1.00000
	1.00000 0.02683	1.00000 0.96464	1.00000 0.99964	0.99991 1.00000	0.97541 1.00000
370	1.00000 0.02427	1.00000 0.96412	1.00000 0.99964	0.99771 1.00000	0.90294 1.00000
	1.00000 0.02427	1.00000 0.96367	1.00000 0.99963	0.99993 1.00000	0.97781 1.00000
380	1.00000 0.02195	1.00000 0.96316	1.00000 0.99963	0.99806 1.00000	0.90892 1.00000
	1.00000 0.02195	1.00000 0.96271	1.00000 0.99962	0.99995 1.00000	0.97998 1.00000
390	1.00000 0.01985	1.00000 0.96221	1.00000 0.99962	0.99836 1.00000	0.91454 1.00000
	1.00000 0.01985	1.00000 0.96175	1.00000 0.99961	0.99996 1.00000	0.98194 1.00000
400	1.00000 0.01795	1.00000 0.96126	1.00000 0.99961	0.99861 1.00000	0.91981 1.00000
	1.00000 0.01795	1.00000 0.96079	1.00000 0.99960	0.99997 1.00000	0.98371 1.00000
410	1.00000 0.01623	1.00000 0.96031	1.00000 0.99960	0.99882 1.00000	0.92475 1.00000
	1.00000 0.01623	1.00000 0.95983	1.00000 0.99959	0.99998 1.00000	0.98530 1.00000
420	1.00000 0.01468	1.00000 0.95936	1.00000 0.99959	0.99900 1.00000	0.92939 1.00000
	1.00000 0.01468	1.00000 0.95887	1.00000 0.99958	0.99998 1.00000	0.98674 1.00000
430	1.00000 0.01328	1.00000 0.95841	1.00000 0.99958	0.99915 1.00000	0.93374 1.00000
	1.00000 0.01328	1.00000 0.95791	1.00000 0.99957	0.99999 1.00000	0.98804 1.00000
440	1.00000 0.01201	1.00000 0.95746	1.00000 0.99957	0.99928 1.00000	0.93783 1.00000
	1.00000 0.01201	1.00000 0.95695	1.00000 0.99956	0.99999 1.00000	0.98921 1.00000
450	1.00000 0.01086	1.00000 0.95651	1.00000 0.99956	0.99939 1.00000	0.94166 1.00000
	1.00000 0.01086	1.00000 0.95600	1.00000 0.99955	0.99999 1.00000	0.99026 1.00000
460	1.00000 0.00982	1.00000 0.95556	1.00000 0.99955	0.99948 1.00000	0.94526 1.00000
	1.00000 0.00982	1.00000 0.95504	1.00000 0.99954	0.99999 1.00000	0.99121 1.00000
470	1.00000 0.00888	1.00000 0.95462	1.00000 0.99954	0.99956 1.00000	0.94863 1.00000
	1.00000 0.00888	1.00000 0.95409	1.00000 0.99953	0.99999 1.00000	0.99207 1.00000
480	1.00000 0.00803	1.00000 0.95367	1.00000 0.99953	0.99963 1.00000	0.95180 1.00000
	1.00000 0.00803	1.00000 0.95313	1.00000 0.99952	1.00000 1.00000	0.99285 1.00000
490	1.00000 0.00727	1.00000 0.95273	1.00000 0.99952	0.99969 1.00000	0.95477 1.00000
	1.00000 0.00727	1.00000 0.95218	1.00000 0.99951	1.00000 1.00000	0.99355 1.00000
500	1.00000 0.00657	1.00000 0.95179	1.00000 0.99951	0.99973 1.00000	0.95756 1.00000
	1.00000 0.00657	1.00000 0.95123	1.00000 0.99950	1.00000 1.00000	0.99418 1.00000
510	1.00000 0.00594	1.00000 0.95084	1.00000 0.99950	0.99977 0.99999	0.96017 1.00000
	1.00000 0.00594	1.00000 0.95028	1.00000 0.99949	1.00000 0.99999	0.99475 1.00000
520	1.00000 0.00537	1.00000 0.94990	1.00000 0.99949	0.99981 0.99999	0.96263 1.00000
	1.00000 0.00537	1.00000 0.94933	1.00000 0.99948	1.00000 0.99999	0.99526 1.00000
530	1.00000 0.00486	1.00000 0.94896	1.00000 0.99948	0.99984 0.99999	0.96493 1.00000
	1.00000 0.00486	1.00000 0.94838	1.00000 0.99947	1.00000 0.99999	0.99573 1.00000

540	1.00000	0.00440	1.00000	0.94802	1.00000	0.99947	0.99986	0.99999	0.96709	1.00000
	1.00000	0.00440	1.00000	0.94743	1.00000	0.99946	1.00000	0.99999	0.99614	1.00000
550	1.00000	0.00398	1.00000	0.94709	1.00000	0.99946	0.99988	0.99999	0.96912	1.00000
	1.00000	0.00398	1.00000	0.94648	1.00000	0.99945	1.00000	0.99999	0.99652	1.00000
560	1.00000	0.00360	1.00000	0.94615	1.00000	0.99945	0.99990	0.99999	0.97103	1.00000
	1.00000	0.00360	1.00000	0.94554	1.00000	0.99944	1.00000	0.99999	0.99686	1.00000
570	1.00000	0.00325	1.00000	0.94521	1.00000	0.99944	0.99992	0.99999	0.97281	1.00000
	1.00000	0.00325	1.00000	0.94459	1.00000	0.99943	1.00000	0.99999	0.99717	1.00000
580	1.00000	0.00294	1.00000	0.94428	1.00000	0.99943	0.99993	0.99999	0.97449	1.00000
	1.00000	0.00294	1.00000	0.94365	1.00000	0.99942	1.00000	0.99999	0.99745	1.00000
590	1.00000	0.00266	1.00000	0.94334	1.00000	0.99942	0.99994	0.99999	0.97606	1.00000
	1.00000	0.00266	1.00000	0.94270	1.00000	0.99941	1.00000	0.99999	0.99770	1.00000
600	1.00000	0.00241	1.00000	0.94241	1.00000	0.99941	0.99995	0.99999	0.97754	1.00000
	1.00000	0.00241	1.00000	0.94176	1.00000	0.99940	1.00000	0.99999	0.99792	1.00000
610	1.00000	0.00218	1.00000	0.94148	1.00000	0.99940	0.99996	0.99999	0.97892	1.00000
	1.00000	0.00218	1.00000	0.94082	1.00000	0.99939	1.00000	0.99999	0.99812	1.00000
620	1.00000	0.00197	1.00000	0.94054	1.00000	0.99939	0.99996	0.99999	0.98022	1.00000
	1.00000	0.00197	1.00000	0.93988	1.00000	0.99938	1.00000	0.99999	0.99831	1.00000
630	1.00000	0.00178	1.00000	0.93961	1.00000	0.99938	0.99997	0.99999	0.98144	1.00000
	1.00000	0.00178	1.00000	0.93894	1.00000	0.99937	1.00000	0.99999	0.99847	1.00000
640	1.00000	0.00161	1.00000	0.93868	1.00000	0.99937	0.99997	0.99999	0.98258	1.00000
	1.00000	0.00161	1.00000	0.93800	1.00000	0.99936	1.00000	0.99999	0.99862	1.00000
650	1.00000	0.00146	1.00000	0.93775	1.00000	0.99936	0.99998	0.99999	0.98366	1.00000
	1.00000	0.00146	1.00000	0.93706	1.00000	0.99935	1.00000	0.99999	0.99876	1.00000
660	1.00000	0.00132	1.00000	0.93683	1.00000	0.99935	0.99998	0.99999	0.98467	1.00000
	1.00000	0.00132	1.00000	0.93613	1.00000	0.99934	1.00000	0.99999	0.99888	1.00000
670	1.00000	0.00119	1.00000	0.93590	1.00000	0.99934	0.99998	0.99999	0.98561	1.00000
	1.00000	0.00119	1.00000	0.93519	1.00000	0.99933	1.00000	0.99999	0.99899	1.00000
680	1.00000	0.00108	1.00000	0.93497	1.00000	0.99933	0.99999	0.99999	0.98650	1.00000
	1.00000	0.00108	1.00000	0.93426	1.00000	0.99932	1.00000	0.99999	0.99909	1.00000
690	1.00000	0.00097	1.00000	0.93405	1.00000	0.99932	0.99999	0.99999	0.98733	1.00000
	1.00000	0.00097	1.00000	0.93332	1.00000	0.99931	1.00000	0.99999	0.99918	1.00000
700	1.00000	0.00088	1.00000	0.93312	1.00000	0.99931	0.99999	0.99999	0.98811	1.00000
	1.00000	0.00088	1.00000	0.93239	1.00000	0.99930	1.00000	0.99999	0.99926	1.00000
710	1.00000	0.00080	1.00000	0.93220	1.00000	0.99930	0.99999	0.99999	0.98884	1.00000
	1.00000	0.00080	1.00000	0.93146	1.00000	0.99929	1.00000	0.99999	0.99933	1.00000
720	1.00000	0.00072	1.00000	0.93128	1.00000	0.99929	0.99999	0.99999	0.98953	1.00000
	1.00000	0.00072	1.00000	0.93053	1.00000	0.99928	1.00000	0.99999	0.99940	1.00000
730	1.00000	0.00065	1.00000	0.93035	1.00000	0.99928	0.99999	0.99999	0.99018	1.00000
	1.00000	0.00065	1.00000	0.92960	1.00000	0.99927	1.00000	0.99999	0.99945	1.00000
740	1.00000	0.00059	1.00000	0.92943	1.00000	0.99927	0.99999	0.99999	0.99078	1.00000
	1.00000	0.00059	1.00000	0.92867	1.00000	0.99926	1.00000	0.99999	0.99951	1.00000
750	1.00000	0.00053	1.00000	0.92851	1.00000	0.99926	1.00000	0.99999	0.99135	1.00000
	1.00000	0.00053	1.00000	0.92774	1.00000	0.99925	1.00000	0.99999	0.99956	1.00000
760	1.00000	0.00048	1.00000	0.92759	1.00000	0.99925	1.00000	0.99999	0.99188	1.00000
	1.00000	0.00048	1.00000	0.92681	1.00000	0.99924	1.00000	0.99999	0.99960	1.00000
770	1.00000	0.00044	1.00000	0.92668	1.00000	0.99924	1.00000	0.99999	0.99238	1.00000
	1.00000	0.00044	1.00000	0.92589	1.00000	0.99923	1.00000	0.99999	0.99964	1.00000
780	1.00000	0.00039	1.00000	0.92576	1.00000	0.99923	1.00000	0.99999	0.99285	1.00000
	1.00000	0.00039	1.00000	0.92496	1.00000	0.99922	1.00000	0.99999	0.99967	1.00000
790	1.00000	0.00036	1.00000	0.92484	1.00000	0.99922	1.00000	0.99999	0.99329	1.00000
	1.00000	0.00036	1.00000	0.92404	1.00000	0.99921	1.00000	0.99999	0.99971	1.00000
800	1.00000	0.00032	1.00000	0.92393	1.00000	0.99921	1.00000	0.99999	0.99371	1.00000
	1.00000	0.00032	1.00000	0.92311	1.00000	0.99920	1.00000	0.99999	0.99973	1.00000
810	1.00000	0.00029	1.00000	0.92301	1.00000	0.99920	1.00000	0.99999	0.99410	1.00000
	1.00000	0.00029	1.00000	0.92219	1.00000	0.99919	1.00000	0.99999	0.99976	1.00000
820	1.00000	0.00026	1.00000	0.92210	1.00000	0.99919	1.00000	0.99999	0.99446	1.00000
	1.00000	0.00026	1.00000	0.92127	1.00000	0.99918	1.00000	0.99999	0.99978	1.00000
830	1.00000	0.00024	1.00000	0.92119	1.00000	0.99918	1.00000	0.99999	0.99480	1.00000
	1.00000	0.00024	1.00000	0.92035	1.00000	0.99917	1.00000	0.99999	0.99981	1.00000
840	1.00000	0.00022	1.00000	0.92028	1.00000	0.99917	1.00000	0.99999	0.99512	1.00000
	1.00000	0.00022	1.00000	0.91943	1.00000	0.99916	1.00000	0.99999	0.99982	1.00000
850	1.00000	0.00019	1.00000	0.91936	1.00000	0.99916	1.00000	0.99999	0.99542	1.00000
	1.00000	0.00019	1.00000	0.91851	1.00000	0.99915	1.00000	0.99999	0.99984	1.00000

860	1.00000	0.00018	1.00000	0.91845	1.00000	0.99915	1.00000	0.99999	0.99570	1.00000
	1.00000	0.00018	1.00000	0.91759	1.00000	0.99914	1.00000	0.99999	0.99986	1.00000
870	1.00000	0.00016	1.00000	0.91755	1.00000	0.99914	1.00000	0.99999	0.99597	1.00000
	1.00000	0.00016	1.00000	0.91667	1.00000	0.99913	1.00000	0.99999	0.99987	1.00000
880	1.00000	0.00014	1.00000	0.91664	1.00000	0.99913	1.00000	0.99999	0.99622	1.00000
	1.00000	0.00014	1.00000	0.91576	1.00000	0.99912	1.00000	0.99999	0.99988	1.00000
890	1.00000	0.00013	1.00000	0.91573	1.00000	0.99912	1.00000	0.99999	0.99645	1.00000
	1.00000	0.00013	1.00000	0.91484	1.00000	0.99911	1.00000	0.99999	0.99989	1.00000
900	1.00000	0.00012	1.00000	0.91482	1.00000	0.99911	1.00000	0.99999	0.99667	1.00000
	1.00000	0.00012	1.00000	0.91393	1.00000	0.99910	1.00000	0.99999	0.99991	1.00000
910	1.00000	0.00011	1.00000	0.91392	1.00000	0.99910	1.00000	0.99999	0.99688	1.00000
	1.00000	0.00011	1.00000	0.91301	1.00000	0.99909	1.00000	0.99999	0.99991	1.00000
920	1.00000	0.00010	1.00000	0.91301	1.00000	0.99909	1.00000	0.99999	0.99707	1.00000
	1.00000	0.00010	1.00000	0.91210	1.00000	0.99908	1.00000	0.99999	0.99992	1.00000
930	1.00000	0.00009	1.00000	0.91211	1.00000	0.99908	1.00000	0.99999	0.99725	1.00000
	1.00000	0.00009	1.00000	0.91119	1.00000	0.99907	1.00000	0.99999	0.99993	1.00000
940	1.00000	0.00008	1.00000	0.91121	1.00000	0.99907	1.00000	0.99999	0.99742	1.00000
	1.00000	0.00008	1.00000	0.91028	1.00000	0.99906	1.00000	0.99999	0.99994	1.00000
950	1.00000	0.00007	1.00000	0.91031	1.00000	0.99906	1.00000	0.99999	0.99758	1.00000
	1.00000	0.00007	1.00000	0.90937	1.00000	0.99905	1.00000	0.99999	0.99994	1.00000
960	1.00000	0.00006	1.00000	0.90940	1.00000	0.99905	1.00000	0.99999	0.99773	1.00000
	1.00000	0.00006	1.00000	0.90846	1.00000	0.99904	1.00000	0.99999	0.99995	1.00000
970	1.00000	0.00006	1.00000	0.90850	1.00000	0.99904	1.00000	0.99999	0.99787	1.00000
	1.00000	0.00006	1.00000	0.90755	1.00000	0.99903	1.00000	0.99999	0.99995	1.00000
980	1.00000	0.00005	1.00000	0.90761	1.00000	0.99903	1.00000	0.99999	0.99800	1.00000
	1.00000	0.00005	1.00000	0.90664	1.00000	0.99902	1.00000	0.99999	0.99996	1.00000
990	1.00000	0.00005	1.00000	0.90671	1.00000	0.99902	1.00000	0.99999	0.99812	1.00000
	1.00000	0.00005	1.00000	0.90574	1.00000	0.99901	1.00000	0.99999	0.99996	1.00000
1000	1.00000	0.00004	1.00000	0.90581	1.00000	0.99901	1.00000	0.99999	0.99824	1.00000
	1.00000	0.00004	1.00000	0.90483	1.00000	0.99900	1.00000	0.99999	0.99997	1.00000

********** p a r a m e t e r s **********

value for r : 0.990000
value for s : 0.100000
value for m_max : 100000
value for k_max : 5
****

The resulting table contains in every field four values

s_net r_net for security/reliability of the MIX-network
s_com r_com as above but of a series-parallel circuit

the horizontal index is for values of 'k'; the vertical index is for values of 'm'

	1		2		3		4		5	
1	0.10000	0.99000	0.00000	1.00000	0.00000	1.00000	0.00000	1.00000	0.00000	1.00000
	0.10000	0.99000	0.00000	1.00000	0.00000	1.00000	0.00000	1.00000	0.00000	1.00000
2	0.19000	0.98010	0.01000	0.99990	0.00000	1.00000	0.00000	1.00000	0.00000	1.00000
	0.19000	0.98010	0.01990	0.99980	0.00000	1.00000	0.00000	1.00000	0.00000	1.00000
3	0.27100	0.97030	0.01900	0.99980	0.00100	1.00000	0.00000	1.00000	0.00000	1.00000
	0.27100	0.97030	0.02970	0.99970	0.00300	1.00000	0.00000	1.00000	0.00000	1.00000
4	0.34390	0.96060	0.02800	0.99970	0.00190	1.00000	0.00010	1.00000	0.00000	1.00000
	0.34390	0.96060	0.03940	0.99960	0.00399	1.00000	0.00040	1.00000	0.00000	1.00000
5	0.40951	0.95099	0.03691	0.99960	0.00280	1.00000	0.00019	1.00000	0.00001	1.00000
	0.40951	0.95099	0.04901	0.99950	0.00499	1.00000	0.00050	1.00000	0.00005	1.00000
6	0.46856	0.94148	0.04574	0.99950	0.00370	1.00000	0.00028	1.00000	0.00002	1.00000
	0.46856	0.94148	0.05852	0.99940	0.00599	0.99999	0.00060	1.00000	0.00006	1.00000
7	0.52170	0.93207	0.05449	0.99941	0.00460	1.00000	0.00037	1.00000	0.00003	1.00000
	0.52170	0.93207	0.06793	0.99930	0.00698	0.99999	0.00070	1.00000	0.00007	1.00000
8	0.56953	0.92274	0.06315	0.99931	0.00550	0.99999	0.00046	1.00000	0.00004	1.00000
	0.56953	0.92274	0.07726	0.99920	0.00797	0.99999	0.00080	1.00000	0.00008	1.00000
9	0.61258	0.91352	0.07174	0.99921	0.00639	0.99999	0.00055	1.00000	0.00005	1.00000
	0.61258	0.91352	0.08648	0.99910	0.00896	0.99999	0.00090	1.00000	0.00009	1.00000
10	0.65132	0.90438	0.08025	0.99911	0.00729	0.99999	0.00064	1.00000	0.00006	1.00000
	0.65132	0.90438	0.09562	0.99900	0.00996	0.99999	0.00100	1.00000	0.00010	1.00000

11	0.68619	0.89534	0.08868	0.99901	0.00819	0.99999	0.00073	1.00000	0.00006	1.00000
	0.68619	0.89534	0.10466	0.99890	0.01095	0.99999	0.00110	1.00000	0.00011	1.00000
12	0.71757	0.88638	0.09704	0.99891	0.00908	0.99999	0.00082	1.00000	0.00007	1.00000
	0.71757	0.88638	0.11362	0.99880	0.01193	0.99999	0.00120	1.00000	0.00012	1.00000
13	0.74581	0.87752	0.10532	0.99881	0.00998	0.99999	0.00091	1.00000	0.00008	1.00000
	0.74581	0.87752	0.12248	0.99870	0.01292	0.99999	0.00130	1.00000	0.00013	1.00000
14	0.77123	0.86875	0.11352	0.99871	0.01087	0.99999	0.00100	1.00000	0.00009	1.00000
	0.77123	0.86875	0.13125	0.99860	0.01391	0.99999	0.00140	1.00000	0.00014	1.00000
15	0.79411	0.86006	0.12164	0.99861	0.01176	0.99999	0.00109	1.00000	0.00010	1.00000
	0.79411	0.86006	0.13994	0.99850	0.01490	0.99999	0.00150	1.00000	0.00015	1.00000
16	0.81470	0.85146	0.12970	0.99851	0.01265	0.99999	0.00118	1.00000	0.00011	1.00000
	0.81470	0.85146	0.14854	0.99840	0.01588	0.99998	0.00160	1.00000	0.00016	1.00000
17	0.83323	0.84294	0.13768	0.99842	0.01355	0.99999	0.00127	1.00000	0.00012	1.00000
	0.83323	0.84294	0.15706	0.99830	0.01686	0.99998	0.00170	1.00000	0.00017	1.00000
18	0.84991	0.83451	0.14558	0.99832	0.01444	0.99998	0.00136	1.00000	0.00013	1.00000
	0.84991	0.83451	0.16549	0.99820	0.01785	0.99998	0.00180	1.00000	0.00018	1.00000
19	0.86491	0.82617	0.15341	0.99822	0.01533	0.99998	0.00145	1.00000	0.00014	1.00000
	0.86491	0.82617	0.17383	0.99810	0.01883	0.99998	0.00190	1.00000	0.00019	1.00000
20	0.87842	0.81791	0.16117	0.99812	0.01621	0.99998	0.00154	1.00000	0.00014	1.00000
	0.87842	0.81791	0.18209	0.99800	0.01981	0.99998	0.00200	1.00000	0.00020	1.00000
21	0.89058	0.80973	0.16886	0.99802	0.01710	0.99998	0.00163	1.00000	0.00015	1.00000
	0.89058	0.80973	0.19027	0.99790	0.02079	0.99998	0.00210	1.00000	0.00021	1.00000
22	0.90152	0.80163	0.17648	0.99792	0.01799	0.99998	0.00172	1.00000	0.00016	1.00000
	0.90152	0.80163	0.19837	0.99780	0.02177	0.99998	0.00220	1.00000	0.00022	1.00000
23	0.91137	0.79361	0.18403	0.99782	0.01887	0.99998	0.00181	1.00000	0.00017	1.00000
	0.91137	0.79361	0.20639	0.99770	0.02275	0.99998	0.00230	1.00000	0.00023	1.00000
24	0.92023	0.78568	0.19151	0.99772	0.01976	0.99998	0.00190	1.00000	0.00018	1.00000
	0.92023	0.78568	0.21432	0.99760	0.02373	0.99998	0.00240	1.00000	0.00024	1.00000
25	0.92821	0.77782	0.19892	0.99763	0.02064	0.99998	0.00199	1.00000	0.00019	1.00000
	0.92821	0.77782	0.22218	0.99750	0.02470	0.99998	0.00250	1.00000	0.00025	1.00000
26	0.93539	0.77004	0.20627	0.99753	0.02153	0.99998	0.00208	1.00000	0.00020	1.00000
	0.93539	0.77004	0.22996	0.99740	0.02568	0.99997	0.00260	1.00000	0.00026	1.00000
27	0.94185	0.76234	0.21354	0.99743	0.02241	0.99998	0.00217	1.00000	0.00021	1.00000
	0.94185	0.76234	0.23766	0.99730	0.02665	0.99997	0.00270	1.00000	0.00027	1.00000
28	0.94767	0.75472	0.22075	0.99733	0.02329	0.99997	0.00226	1.00000	0.00022	1.00000
	0.94767	0.75472	0.24528	0.99720	0.02763	0.99997	0.00280	1.00000	0.00028	1.00000
29	0.95290	0.74717	0.22790	0.99723	0.02418	0.99997	0.00235	1.00000	0.00023	1.00000
	0.95290	0.74717	0.25283	0.99710	0.02860	0.99997	0.00290	1.00000	0.00029	1.00000
30	0.95761	0.73970	0.23498	0.99713	0.02506	0.99997	0.00244	1.00000	0.00023	1.00000
	0.95761	0.73970	0.26030	0.99700	0.02957	0.99997	0.00300	1.00000	0.00030	1.00000
31	0.96185	0.73230	0.24199	0.99703	0.02594	0.99997	0.00253	1.00000	0.00024	1.00000
	0.96185	0.73230	0.26770	0.99690	0.03054	0.99997	0.00310	1.00000	0.00031	1.00000
32	0.96566	0.72498	0.24894	0.99693	0.02681	0.99997	0.00262	1.00000	0.00025	1.00000
	0.96566	0.72498	0.27502	0.99680	0.03151	0.99997	0.00320	1.00000	0.00032	1.00000
33	0.96910	0.71773	0.25582	0.99684	0.02769	0.99997	0.00271	1.00000	0.00026	1.00000
	0.96910	0.71773	0.28227	0.99671	0.03248	0.99997	0.00329	1.00000	0.00033	1.00000
34	0.97219	0.71055	0.26265	0.99674	0.02857	0.99997	0.00280	1.00000	0.00027	1.00000
	0.97219	0.71055	0.28945	0.99661	0.03344	0.99997	0.00339	1.00000	0.00034	1.00000
35	0.97497	0.70345	0.26941	0.99664	0.02945	0.99997	0.00289	1.00000	0.00028	1.00000
	0.97497	0.70345	0.29655	0.99651	0.03441	0.99997	0.00349	1.00000	0.00035	1.00000
36	0.97747	0.69641	0.27610	0.99654	0.03032	0.99997	0.00298	1.00000	0.00029	1.00000
	0.97747	0.69641	0.30359	0.99641	0.03538	0.99996	0.00359	1.00000	0.00036	1.00000
37	0.97972	0.68945	0.28274	0.99644	0.03120	0.99997	0.00307	1.00000	0.00030	1.00000
	0.97972	0.68945	0.31055	0.99631	0.03634	0.99996	0.00369	1.00000	0.00037	1.00000
38	0.98175	0.68255	0.28931	0.99634	0.03207	0.99996	0.00316	1.00000	0.00031	1.00000
	0.98175	0.68255	0.31745	0.99621	0.03731	0.99996	0.00379	1.00000	0.00038	1.00000
39	0.98358	0.67573	0.29583	0.99624	0.03295	0.99996	0.00325	1.00000	0.00032	1.00000
	0.98358	0.67573	0.32427	0.99611	0.03827	0.99996	0.00389	1.00000	0.00039	1.00000
40	0.98522	0.66897	0.30228	0.99614	0.03382	0.99996	0.00334	1.00000	0.00032	1.00000
	0.98522	0.66897	0.33103	0.99601	0.03923	0.99996	0.00399	1.00000	0.00040	1.00000
41	0.98670	0.66228	0.30868	0.99605	0.03469	0.99996	0.00343	1.00000	0.00033	1.00000
	0.98670	0.66228	0.33772	0.99591	0.04019	0.99996	0.00409	1.00000	0.00041	1.00000
42	0.98803	0.65566	0.31502	0.99595	0.03556	0.99996	0.00352	1.00000	0.00034	1.00000
	0.98803	0.65566	0.34434	0.99581	0.04115	0.99996	0.00419	1.00000	0.00042	1.00000

43	0.98922 0.64910	0.32130 0.99585	0.03643 0.99996	0.00360 1.00000	0.00035 1.00000
	0.98922 0.64910	0.35090 0.99571	0.04211 0.99996	0.00429 1.00000	0.00043 1.00000
44	0.99030 0.64261	0.32752 0.99575	0.03730 0.99996	0.00369 1.00000	0.00036 1.00000
	0.99030 0.64261	0.35739 0.99561	0.04307 0.99996	0.00439 1.00000	0.00044 1.00000
45	0.99127 0.63619	0.33368 0.99565	0.03817 0.99996	0.00378 1.00000	0.00037 1.00000
	0.99127 0.63619	0.36381 0.99551	0.04402 0.99996	0.00449 1.00000	0.00045 1.00000
46	0.99214 0.62982	0.33979 0.99555	0.03904 0.99996	0.00387 1.00000	0.00038 1.00000
	0.99214 0.62982	0.37018 0.99541	0.04498 0.99995	0.00459 1.00000	0.00046 1.00000
47	0.99293 0.62353	0.34585 0.99545	0.03991 0.99996	0.00396 1.00000	0.00039 1.00000
	0.99293 0.62353	0.37647 0.99531	0.04594 0.99995	0.00469 1.00000	0.00047 1.00000
48	0.99364 0.61729	0.35184 0.99536	0.04077 0.99995	0.00405 1.00000	0.00040 1.00000
	0.99364 0.61729	0.38271 0.99521	0.04689 0.99995	0.00479 1.00000	0.00048 1.00000
49	0.99427 0.61112	0.35778 0.99526	0.04164 0.99995	0.00414 1.00000	0.00041 1.00000
	0.99427 0.61112	0.38888 0.99511	0.04784 0.99995	0.00489 1.00000	0.00049 1.00000
50	0.99485 0.60501	0.36367 0.99516	0.04250 0.99995	0.00423 1.00000	0.00041 1.00000
	0.99485 0.60501	0.39499 0.99501	0.04879 0.99995	0.00499 1.00000	0.00050 1.00000
51	0.99536 0.59896	0.36950 0.99506	0.04337 0.99995	0.00432 1.00000	0.00042 1.00000
	0.99536 0.59896	0.40104 0.99491	0.04975 0.99995	0.00509 1.00000	0.00051 1.00000
52	0.99583 0.59297	0.37528 0.99496	0.04423 0.99995	0.00441 1.00000	0.00043 1.00000
	0.99583 0.59297	0.40703 0.99481	0.05070 0.99995	0.00519 1.00000	0.00052 1.00000
53	0.99624 0.58704	0.38101 0.99486	0.04509 0.99995	0.00450 1.00000	0.00044 1.00000
	0.99624 0.58704	0.41296 0.99471	0.05165 0.99995	0.00529 1.00000	0.00053 1.00000
54	0.99662 0.58117	0.38669 0.99476	0.04595 0.99995	0.00459 1.00000	0.00045 1.00000
	0.99662 0.58117	0.41883 0.99461	0.05259 0.99995	0.00539 1.00000	0.00054 1.00000
55	0.99696 0.57535	0.39231 0.99467	0.04682 0.99995	0.00468 1.00000	0.00046 1.00000
	0.99696 0.57535	0.42465 0.99451	0.05354 0.99995	0.00549 1.00000	0.00055 1.00000
56	0.99726 0.56960	0.39788 0.99457	0.04768 0.99995	0.00477 1.00000	0.00047 1.00000
	0.99726 0.56960	0.43040 0.99442	0.05449 0.99994	0.00558 1.00000	0.00056 1.00000
57	0.99753 0.56391	0.40340 0.99447	0.04853 0.99995	0.00486 1.00000	0.00048 1.00000
	0.99753 0.56391	0.43609 0.99432	0.05543 0.99994	0.00568 1.00000	0.00057 1.00000
58	0.99778 0.55827	0.40887 0.99437	0.04939 0.99994	0.00495 1.00000	0.00049 1.00000
	0.99778 0.55827	0.44173 0.99422	0.05638 0.99994	0.00578 1.00000	0.00058 1.00000
59	0.99800 0.55268	0.41429 0.99427	0.05025 0.99994	0.00504 1.00000	0.00050 1.00000
	0.99800 0.55268	0.44732 0.99412	0.05732 0.99994	0.00588 1.00000	0.00059 1.00000
60	0.99820 0.54716	0.41966 0.99417	0.05111 0.99994	0.00513 1.00000	0.00050 1.00000
	0.99820 0.54716	0.45284 0.99402	0.05826 0.99994	0.00598 1.00000	0.00060 1.00000
61	0.99838 0.54169	0.42498 0.99408	0.05196 0.99994	0.00522 1.00000	0.00051 1.00000
	0.99838 0.54169	0.45831 0.99392	0.05921 0.99994	0.00608 1.00000	0.00061 1.00000
62	0.99854 0.53627	0.43025 0.99398	0.05282 0.99994	0.00531 1.00000	0.00052 1.00000
	0.99854 0.53627	0.46373 0.99382	0.06015 0.99994	0.00618 1.00000	0.00062 1.00000
63	0.99869 0.53091	0.43547 0.99388	0.05368 0.99994	0.00540 1.00000	0.00053 1.00000
	0.99869 0.53091	0.46909 0.99372	0.06109 0.99994	0.00628 1.00000	0.00063 1.00000
64	0.99882 0.52560	0.44065 0.99378	0.05453 0.99994	0.00549 1.00000	0.00054 1.00000
	0.99882 0.52560	0.47440 0.99362	0.06203 0.99994	0.00638 1.00000	0.00064 1.00000
65	0.99894 0.52034	0.44577 0.99368	0.05538 0.99994	0.00558 1.00000	0.00055 1.00000
	0.99894 0.52034	0.47966 0.99352	0.06296 0.99994	0.00648 1.00000	0.00065 1.00000
66	0.99904 0.51514	0.45086 0.99358	0.05623 0.99994	0.00567 1.00000	0.00056 1.00000
	0.99904 0.51514	0.48486 0.99342	0.06390 0.99993	0.00658 1.00000	0.00066 1.00000
67	0.99914 0.50999	0.45589 0.99348	0.05709 0.99994	0.00576 1.00000	0.00057 1.00000
	0.99914 0.50999	0.49001 0.99332	0.06484 0.99993	0.00668 1.00000	0.00067 1.00000
68	0.99923 0.50489	0.46088 0.99339	0.05794 0.99993	0.00585 1.00000	0.00058 1.00000
	0.99923 0.50489	0.49511 0.99322	0.06577 0.99993	0.00678 1.00000	0.00068 1.00000
69	0.99930 0.49984	0.46582 0.99329	0.05879 0.99993	0.00593 1.00000	0.00059 1.00000
	0.99930 0.49984	0.50016 0.99312	0.06671 0.99993	0.00688 1.00000	0.00069 1.00000
70	0.99937 0.49484	0.47072 0.99319	0.05964 0.99993	0.00602 1.00000	0.00059 1.00000
	0.99937 0.49484	0.50516 0.99302	0.06764 0.99993	0.00698 1.00000	0.00070 1.00000
71	0.99944 0.48989	0.47557 0.99309	0.06049 0.99993	0.00611 1.00000	0.00060 1.00000
	0.99944 0.48989	0.51011 0.99292	0.06857 0.99993	0.00708 1.00000	0.00071 1.00000
72	0.99949 0.48499	0.48038 0.99299	0.06133 0.99993	0.00620 1.00000	0.00061 1.00000
	0.99949 0.48499	0.51501 0.99283	0.06950 0.99993	0.00717 1.00000	0.00072 1.00000
73	0.99954 0.48014	0.48514 0.99289	0.06218 0.99993	0.00629 1.00000	0.00062 1.00000
	0.99954 0.48014	0.51986 0.99273	0.07043 0.99993	0.00727 1.00000	0.00073 1.00000
74	0.99959 0.47534	0.48986 0.99280	0.06303 0.99993	0.00638 1.00000	0.00063 1.00000
	0.99959 0.47534	0.52466 0.99263	0.07136 0.99993	0.00737 1.00000	0.00074 1.00000

75	0.99963 0.47059	0.49454 0.99270	0.06387 0.99993	0.00647 1.00000	0.00064 1.00000
	0.99963 0.47059	0.52941 0.99253	0.07229 0.99993	0.00747 1.00000	0.00075 1.00000
76	0.99967 0.46588	0.49917 0.99260	0.06472 0.99993	0.00656 1.00000	0.00065 1.00000
	0.99967 0.46588	0.53412 0.99243	0.07322 0.99992	0.00757 1.00000	0.00076 1.00000
77	0.99970 0.46122	0.50376 0.99250	0.06556 0.99993	0.00665 1.00000	0.00066 1.00000
	0.99970 0.46122	0.53878 0.99233	0.07415 0.99992	0.00767 1.00000	0.00077 1.00000
78	0.99973 0.45661	0.50831 0.99240	0.06640 0.99992	0.00674 1.00000	0.00067 1.00000
	0.99973 0.45661	0.54339 0.99223	0.07507 0.99992	0.00777 1.00000	0.00078 1.00000
79	0.99976 0.45204	0.51282 0.99230	0.06725 0.99992	0.00683 1.00000	0.00068 1.00000
	0.99976 0.45204	0.54796 0.99213	0.07600 0.99992	0.00787 1.00000	0.00079 1.00000
80	0.99978 0.44752	0.51728 0.99221	0.06809 0.99992	0.00692 1.00000	0.00068 1.00000
	0.99978 0.44752	0.55248 0.99203	0.07692 0.99992	0.00797 1.00000	0.00080 1.00000
81	0.99980 0.44305	0.52171 0.99211	0.06893 0.99992	0.00701 1.00000	0.00069 1.00000
	0.99980 0.44305	0.55695 0.99193	0.07784 0.99992	0.00807 1.00000	0.00081 1.00000
82	0.99982 0.43862	0.52609 0.99201	0.06977 0.99992	0.00710 1.00000	0.00070 1.00000
	0.99982 0.43862	0.56138 0.99183	0.07877 0.99992	0.00817 1.00000	0.00082 1.00000
83	0.99984 0.43423	0.53044 0.99191	0.07061 0.99992	0.00719 1.00000	0.00071 1.00000
	0.99984 0.43423	0.56577 0.99173	0.07969 0.99992	0.00827 1.00000	0.00083 1.00000
84	0.99986 0.42989	0.53474 0.99181	0.07145 0.99992	0.00728 1.00000	0.00072 1.00000
	0.99986 0.42989	0.57011 0.99163	0.08061 0.99992	0.00837 1.00000	0.00084 1.00000
85	0.99987 0.42559	0.53901 0.99172	0.07229 0.99992	0.00737 1.00000	0.00073 1.00000
	0.99987 0.42559	0.57441 0.99154	0.08153 0.99992	0.00846 1.00000	0.00085 1.00000
86	0.99988 0.42133	0.54323 0.99162	0.07312 0.99992	0.00746 1.00000	0.00074 1.00000
	0.99988 0.42133	0.57867 0.99144	0.08245 0.99991	0.00856 1.00000	0.00086 1.00000
87	0.99990 0.41712	0.54742 0.99152	0.07396 0.99992	0.00754 1.00000	0.00075 1.00000
	0.99990 0.41712	0.58288 0.99134	0.08336 0.99991	0.00866 1.00000	0.00087 1.00000
88	0.99991 0.41295	0.55157 0.99142	0.07480 0.99991	0.00763 1.00000	0.00076 1.00000
	0.99991 0.41295	0.58705 0.99124	0.08428 0.99991	0.00876 1.00000	0.00088 1.00000
89	0.99992 0.40882	0.55568 0.99132	0.07563 0.99991	0.00772 1.00000	0.00077 1.00000
	0.99992 0.40882	0.59118 0.99114	0.08520 0.99991	0.00886 1.00000	0.00089 1.00000
90	0.99992 0.40473	0.55975 0.99122	0.07646 0.99991	0.00781 1.00000	0.00077 1.00000
	0.99992 0.40473	0.59527 0.99104	0.08611 0.99991	0.00896 1.00000	0.00090 1.00000
91	0.99993 0.40068	0.56379 0.99113	0.07730 0.99991	0.00790 1.00000	0.00078 1.00000
	0.99993 0.40068	0.59932 0.99094	0.08702 0.99991	0.00906 1.00000	0.00091 1.00000
92	0.99994 0.39668	0.56779 0.99103	0.07813 0.99991	0.00799 1.00000	0.00079 1.00000
	0.99994 0.39668	0.60332 0.99084	0.08794 0.99991	0.00916 1.00000	0.00092 1.00000
93	0.99994 0.39271	0.57175 0.99093	0.07896 0.99991	0.00808 1.00000	0.00080 1.00000
	0.99994 0.39271	0.60729 0.99074	0.08885 0.99991	0.00926 1.00000	0.00093 1.00000
94	0.99995 0.38878	0.57568 0.99083	0.07979 0.99991	0.00817 1.00000	0.00081 1.00000
	0.99995 0.38878	0.61122 0.99064	0.08976 0.99991	0.00936 1.00000	0.00094 1.00000
95	0.99996 0.38490	0.57957 0.99073	0.08062 0.99991	0.00826 1.00000	0.00082 1.00000
	0.99996 0.38490	0.61510 0.99054	0.09067 0.99991	0.00946 1.00000	0.00095 1.00000
96	0.99996 0.38105	0.58342 0.99064	0.08145 0.99991	0.00835 1.00000	0.00083 1.00000
	0.99996 0.38105	0.61895 0.99045	0.09158 0.99990	0.00955 1.00000	0.00096 1.00000
97	0.99996 0.37724	0.58724 0.99054	0.08228 0.99991	0.00844 1.00000	0.00084 1.00000
	0.99996 0.37724	0.62276 0.99035	0.09249 0.99990	0.00965 1.00000	0.00097 1.00000
98	0.99997 0.37346	0.59102 0.99044	0.08311 0.99990	0.00853 1.00000	0.00085 1.00000
	0.99997 0.37346	0.62654 0.99025	0.09340 0.99990	0.00975 1.00000	0.00098 1.00000
99	0.99997 0.36973	0.59477 0.99034	0.08394 0.99990	0.00862 1.00000	0.00086 1.00000
	0.99997 0.36973	0.63027 0.99015	0.09430 0.99990	0.00985 1.00000	0.00099 1.00000
100	0.99997 0.36603	0.59849 0.99024	0.08477 0.99990	0.00871 1.00000	0.00086 1.00000
	0.99997 0.36603	0.63397 0.99005	0.09521 0.99990	0.00995 1.00000	0.00100 1.00000
110	0.99999 0.33103	0.63381 0.98926	0.09299 0.99989	0.00960 1.00000	0.00095 1.00000
	0.99999 0.33103	0.66897 0.98906	0.10422 0.99989	0.01094 1.00000	0.00110 1.00000
120	1.00000 0.29938	0.66603 0.98828	0.10114 0.99988	0.01049 1.00000	0.00104 1.00000
	1.00000 0.29938	0.70062 0.98807	0.11313 0.99988	0.01193 1.00000	0.00120 1.00000
130	1.00000 0.27075	0.69542 0.98731	0.10922 0.99987	0.01138 1.00000	0.00113 1.00000
	1.00000 0.27075	0.72925 0.98708	0.12196 0.99987	0.01292 1.00000	0.00130 1.00000
140	1.00000 0.24487	0.72221 0.98633	0.11723 0.99986	0.01227 1.00000	0.00122 1.00000
	1.00000 0.24487	0.75513 0.98610	0.13070 0.99986	0.01390 1.00000	0.00140 1.00000
150	1.00000 0.22145	0.74665 0.98535	0.12516 0.99985	0.01316 1.00000	0.00131 1.00000
	1.00000 0.22145	0.77855 0.98511	0.13936 0.99985	0.01489 1.00000	0.00150 1.00000
160	1.00000 0.20028	0.76894 0.98438	0.13303 0.99984	0.01405 1.00000	0.00140 1.00000
	1.00000 0.20028	0.79972 0.98413	0.14792 0.99984	0.01587 1.00000	0.00160 1.00000

170	1.00000 0.18113	0.78927 0.98340	0.14082 0.99983	0.01493 1.00000	0.00149 1.00000
	1.00000 0.18113	0.81887 0.98314	0.15641 0.99983	0.01686 1.00000	0.00170 1.00000
180	1.00000 0.16381	0.80781 0.98243	0.14854 0.99982	0.01582 1.00000	0.00158 1.00000
	1.00000 0.16381	0.83619 0.98216	0.16481 0.99982	0.01784 1.00000	0.00180 1.00000
190	1.00000 0.14814	0.82472 0.98146	0.15619 0.99981	0.01671 1.00000	0.00167 1.00000
	1.00000 0.14814	0.85186 0.98118	0.17312 0.99981	0.01882 1.00000	0.00190 1.00000
200	1.00000 0.13398	0.84014 0.98049	0.16378 0.99980	0.01759 1.00000	0.00176 1.00000
	1.00000 0.13398	0.86602 0.98020	0.18135 0.99980	0.01980 1.00000	0.00200 1.00000
210	1.00000 0.12117	0.85421 0.97952	0.17129 0.99979	0.01847 1.00000	0.00185 1.00000
	1.00000 0.12117	0.87883 0.97922	0.18950 0.99979	0.02078 1.00000	0.00210 1.00000
220	1.00000 0.10958	0.86704 0.97855	0.17874 0.99978	0.01936 1.00000	0.00194 1.00000
	1.00000 0.10958	0.89042 0.97824	0.19757 0.99978	0.02176 1.00000	0.00220 1.00000
230	1.00000 0.09910	0.87873 0.97758	0.18612 0.99977	0.02024 1.00000	0.00203 1.00000
	1.00000 0.09910	0.90090 0.97726	0.20556 0.99977	0.02274 1.00000	0.00230 1.00000
240	1.00000 0.08963	0.88940 0.97661	0.19344 0.99976	0.02112 1.00000	0.00212 1.00000
	1.00000 0.08963	0.91037 0.97628	0.21347 0.99976	0.02372 1.00000	0.00240 1.00000
250	1.00000 0.08106	0.89913 0.97564	0.20069 0.99975	0.02200 1.00000	0.00221 1.00000
	1.00000 0.08106	0.91894 0.97531	0.22130 0.99975	0.02469 1.00000	0.00250 1.00000
260	1.00000 0.07331	0.90801 0.97468	0.20787 0.99974	0.02288 1.00000	0.00230 1.00000
	1.00000 0.07331	0.92669 0.97433	0.22905 0.99974	0.02567 1.00000	0.00260 1.00000
270	1.00000 0.06630	0.91610 0.97371	0.21499 0.99973	0.02376 1.00000	0.00239 1.00000
	1.00000 0.06630	0.93370 0.97336	0.23672 0.99973	0.02664 1.00000	0.00270 1.00000
280	1.00000 0.05996	0.92348 0.97275	0.22205 0.99972	0.02464 1.00000	0.00248 1.00000
	1.00000 0.05996	0.94004 0.97239	0.24432 0.99972	0.02761 1.00000	0.00280 1.00000
290	1.00000 0.05423	0.93022 0.97179	0.22904 0.99971	0.02552 1.00000	0.00257 1.00000
	1.00000 0.05423	0.94577 0.97142	0.25184 0.99971	0.02858 1.00000	0.00290 1.00000
300	1.00000 0.04904	0.93636 0.97082	0.23597 0.99971	0.02640 1.00000	0.00266 1.00000
	1.00000 0.04904	0.95096 0.97044	0.25929 0.99970	0.02956 1.00000	0.00300 1.00000
310	1.00000 0.04435	0.94195 0.96986	0.24283 0.99970	0.02727 1.00000	0.00275 1.00000
	1.00000 0.04435	0.95565 0.96947	0.26667 0.99969	0.03053 1.00000	0.00310 1.00000
320	1.00000 0.04011	0.94706 0.96890	0.24964 0.99969	0.02815 1.00000	0.00284 1.00000
	1.00000 0.04011	0.95989 0.96851	0.27397 0.99968	0.03149 1.00000	0.00319 1.00000
330	1.00000 0.03628	0.95172 0.96794	0.25638 0.99968	0.02902 1.00000	0.00293 1.00000
	1.00000 0.03628	0.96372 0.96754	0.28119 0.99967	0.03246 1.00000	0.00329 1.00000
340	1.00000 0.03281	0.95597 0.96699	0.26307 0.99967	0.02990 1.00000	0.00302 1.00000
	1.00000 0.03281	0.96719 0.96657	0.28835 0.99966	0.03343 1.00000	0.00339 1.00000
350	1.00000 0.02967	0.95984 0.96603	0.26969 0.99966	0.03077 1.00000	0.00311 1.00000
	1.00000 0.02967	0.97033 0.96560	0.29544 0.99965	0.03440 1.00000	0.00349 1.00000
360	1.00000 0.02683	0.96337 0.96507	0.27625 0.99965	0.03164 1.00000	0.00320 1.00000
	1.00000 0.02683	0.97317 0.96464	0.30245 0.99964	0.03536 1.00000	0.00359 1.00000
370	1.00000 0.02427	0.96660 0.96412	0.28276 0.99964	0.03251 1.00000	0.00329 1.00000
	1.00000 0.02427	0.97573 0.96367	0.30939 0.99963	0.03633 1.00000	0.00369 1.00000
380	1.00000 0.02195	0.96954 0.96316	0.28921 0.99963	0.03338 1.00000	0.00338 1.00000
	1.00000 0.02195	0.97805 0.96271	0.31627 0.99962	0.03729 1.00000	0.00379 1.00000
390	1.00000 0.01985	0.97222 0.96221	0.29559 0.99962	0.03425 1.00000	0.00347 1.00000
	1.00000 0.01985	0.98015 0.96175	0.32308 0.99961	0.03825 1.00000	0.00389 1.00000
400	1.00000 0.01795	0.97466 0.96126	0.30193 0.99961	0.03512 1.00000	0.00356 1.00000
	1.00000 0.01795	0.98205 0.96079	0.32981 0.99960	0.03921 1.00000	0.00399 1.00000
410	1.00000 0.01623	0.97689 0.96031	0.30820 0.99960	0.03599 1.00000	0.00365 1.00000
	1.00000 0.01623	0.98377 0.95983	0.33649 0.99959	0.04017 1.00000	0.00409 1.00000
420	1.00000 0.01468	0.97892 0.95936	0.31442 0.99959	0.03686 1.00000	0.00374 1.00000
	1.00000 0.01468	0.98532 0.95887	0.34309 0.99958	0.04113 1.00000	0.00419 1.00000
430	1.00000 0.01328	0.98078 0.95841	0.32058 0.99958	0.03773 1.00000	0.00383 1.00000
	1.00000 0.01328	0.98672 0.95791	0.34963 0.99957	0.04209 1.00000	0.00429 1.00000
440	1.00000 0.01201	0.98247 0.95746	0.32669 0.99957	0.03859 1.00000	0.00392 1.00000
	1.00000 0.01201	0.98799 0.95695	0.35611 0.99956	0.04305 1.00000	0.00439 1.00000
450	1.00000 0.01086	0.98401 0.95651	0.33274 0.99956	0.03946 1.00000	0.00401 1.00000
	1.00000 0.01086	0.98914 0.95600	0.36252 0.99955	0.04400 1.00000	0.00449 1.00000
460	1.00000 0.00982	0.98542 0.95556	0.33873 0.99955	0.04032 1.00000	0.00410 1.00000
	1.00000 0.00982	0.99018 0.95504	0.36886 0.99954	0.04496 1.00000	0.00459 1.00000
470	1.00000 0.00888	0.98670 0.95462	0.34468 0.99954	0.04118 1.00000	0.00419 1.00000
	1.00000 0.00888	0.99112 0.95409	0.37514 0.99953	0.04591 1.00000	0.00469 1.00000
480	1.00000 0.00803	0.98787 0.95367	0.35057 0.99953	0.04205 1.00000	0.00428 1.00000
	1.00000 0.00803	0.99197 0.95313	0.38137 0.99952	0.04687 1.00000	0.00479 1.00000

490	1.00000	0.00727	0.98894	0.95273	0.35640	0.99952	0.04291	1.00000	0.00437	1.00000
	1.00000	0.00727	0.99273	0.95218	0.38752	0.99951	0.04782	1.00000	0.00489	1.00000
500	1.00000	0.00657	0.98991	0.95179	0.36219	0.99951	0.04377	1.00000	0.00446	1.00000
	1.00000	0.00657	0.99343	0.95123	0.39362	0.99950	0.04877	1.00000	0.00499	1.00000
510	1.00000	0.00594	0.99080	0.95084	0.36792	0.99950	0.04463	0.99999	0.00454	1.00000
	1.00000	0.00594	0.99406	0.95028	0.39966	0.99949	0.04972	0.99999	0.00509	1.00000
520	1.00000	0.00537	0.99161	0.94990	0.37360	0.99949	0.04549	0.99999	0.00463	1.00000
	1.00000	0.00537	0.99463	0.94933	0.40563	0.99948	0.05067	0.99999	0.00519	1.00000
530	1.00000	0.00486	0.99235	0.94896	0.37923	0.99948	0.04635	0.99999	0.00472	1.00000
	1.00000	0.00486	0.99514	0.94838	0.41155	0.99947	0.05162	0.99999	0.00529	1.00000
540	1.00000	0.00440	0.99302	0.94802	0.38481	0.99947	0.04721	0.99999	0.00481	1.00000
	1.00000	0.00440	0.99560	0.94743	0.41741	0.99946	0.05257	0.99999	0.00539	1.00000
550	1.00000	0.00398	0.99363	0.94709	0.39034	0.99946	0.04807	0.99999	0.00490	1.00000
	1.00000	0.00398	0.99602	0.94648	0.42321	0.99945	0.05352	0.99999	0.00548	1.00000
560	1.00000	0.00360	0.99419	0.94615	0.39582	0.99945	0.04892	0.99999	0.00499	1.00000
	1.00000	0.00360	0.99640	0.94554	0.42895	0.99944	0.05446	0.99999	0.00558	1.00000
570	1.00000	0.00325	0.99471	0.94521	0.40125	0.99944	0.04978	0.99999	0.00508	1.00000
	1.00000	0.00325	0.99675	0.94459	0.43464	0.99943	0.05541	0.99999	0.00568	1.00000
580	1.00000	0.00294	0.99517	0.94428	0.40663	0.99943	0.05063	0.99999	0.00517	1.00000
	1.00000	0.00294	0.99706	0.94365	0.44026	0.99942	0.05635	0.99999	0.00578	1.00000
590	1.00000	0.00266	0.99560	0.94334	0.41197	0.99942	0.05149	0.99999	0.00526	1.00000
	1.00000	0.00266	0.99734	0.94270	0.44584	0.99941	0.05730	0.99999	0.00588	1.00000
600	1.00000	0.00241	0.99598	0.94241	0.41725	0.99941	0.05234	0.99999	0.00535	1.00000
	1.00000	0.00241	0.99759	0.94176	0.45135	0.99940	0.05824	0.99999	0.00598	1.00000
610	1.00000	0.00218	0.99634	0.94148	0.42249	0.99940	0.05319	0.99999	0.00544	1.00000
	1.00000	0.00218	0.99782	0.94082	0.45681	0.99939	0.05918	0.99999	0.00608	1.00000
620	1.00000	0.00197	0.99666	0.94054	0.42768	0.99939	0.05405	0.99999	0.00553	1.00000
	1.00000	0.00197	0.99803	0.93988	0.46222	0.99938	0.06012	0.99999	0.00618	1.00000
630	1.00000	0.00178	0.99695	0.93961	0.43282	0.99938	0.05490	0.99999	0.00562	1.00000
	1.00000	0.00178	0.99822	0.93894	0.46758	0.99937	0.06106	0.99999	0.00628	1.00000
640	1.00000	0.00161	0.99722	0.93868	0.43792	0.99937	0.05575	0.99999	0.00571	1.00000
	1.00000	0.00161	0.99839	0.93800	0.47288	0.99936	0.06200	0.99999	0.00638	1.00000
650	1.00000	0.00146	0.99747	0.93775	0.44297	0.99936	0.05660	0.99999	0.00580	1.00000
	1.00000	0.00146	0.99854	0.93706	0.47812	0.99935	0.06294	0.99999	0.00648	1.00000
660	1.00000	0.00132	0.99769	0.93683	0.44798	0.99935	0.05745	0.99999	0.00589	1.00000
	1.00000	0.00132	0.99868	0.93613	0.48332	0.99934	0.06387	0.99999	0.00658	1.00000
670	1.00000	0.00119	0.99789	0.93590	0.45294	0.99934	0.05830	0.99999	0.00598	1.00000
	1.00000	0.00119	0.99881	0.93519	0.48846	0.99933	0.06481	0.99999	0.00668	1.00000
680	1.00000	0.00108	0.99808	0.93497	0.45786	0.99933	0.05914	0.99999	0.00607	1.00000
	1.00000	0.00108	0.99892	0.93426	0.49356	0.99932	0.06574	0.99999	0.00678	1.00000
690	1.00000	0.00097	0.99825	0.93405	0.46273	0.99932	0.05999	0.99999	0.00616	1.00000
	1.00000	0.00097	0.99903	0.93332	0.49860	0.99931	0.06668	0.99999	0.00688	1.00000
700	1.00000	0.00088	0.99840	0.93312	0.46756	0.99931	0.06084	0.99999	0.00625	1.00000
	1.00000	0.00088	0.99912	0.93239	0.50359	0.99930	0.06761	0.99999	0.00698	1.00000
710	1.00000	0.00080	0.99854	0.93220	0.47234	0.99930	0.06168	0.99999	0.00634	1.00000
	1.00000	0.00080	0.99920	0.93146	0.50853	0.99929	0.06854	0.99999	0.00707	1.00000
720	1.00000	0.00072	0.99867	0.93128	0.47709	0.99929	0.06253	0.99999	0.00642	1.00000
	1.00000	0.00072	0.99928	0.93053	0.51342	0.99928	0.06947	0.99999	0.00717	1.00000
730	1.00000	0.00065	0.99879	0.93035	0.48179	0.99928	0.06337	0.99999	0.00651	1.00000
	1.00000	0.00065	0.99935	0.92960	0.51827	0.99927	0.07040	0.99999	0.00727	1.00000
740	1.00000	0.00059	0.99889	0.92943	0.48644	0.99927	0.06421	0.99999	0.00660	1.00000
	1.00000	0.00059	0.99941	0.92867	0.52306	0.99926	0.07133	0.99999	0.00737	1.00000
750	1.00000	0.00053	0.99899	0.92851	0.49106	0.99926	0.06505	0.99999	0.00669	1.00000
	1.00000	0.00053	0.99947	0.92774	0.52781	0.99925	0.07226	0.99999	0.00747	1.00000
760	1.00000	0.00048	0.99908	0.92759	0.49563	0.99925	0.06590	0.99999	0.00678	1.00000
	1.00000	0.00048	0.99952	0.92681	0.53251	0.99924	0.07319	0.99999	0.00757	1.00000
770	1.00000	0.00044	0.99916	0.92668	0.50017	0.99924	0.06674	0.99999	0.00687	1.00000
	1.00000	0.00044	0.99956	0.92589	0.53717	0.99923	0.07411	0.99999	0.00767	1.00000
780	1.00000	0.00039	0.99923	0.92576	0.50466	0.99923	0.06758	0.99999	0.00696	1.00000
	1.00000	0.00039	0.99961	0.92496	0.54177	0.99922	0.07504	0.99999	0.00777	1.00000
790	1.00000	0.00036	0.99930	0.92484	0.50911	0.99922	0.06842	0.99999	0.00705	1.00000
	1.00000	0.00036	0.99964	0.92404	0.54633	0.99921	0.07596	0.99999	0.00787	1.00000
800	1.00000	0.00032	0.99936	0.92393	0.51352	0.99921	0.06925	0.99999	0.00714	1.00000
	1.00000	0.00032	0.99968	0.92311	0.55085	0.99920	0.07689	0.99999	0.00797	1.00000

810	1.00000 0.00029	0.99942 0.92301	0.51790 0.99920	0.07009 0.99999	0.00723 1.00000
	1.00000 0.00029	0.99971 0.92219	0.55532 0.99919	0.07781 0.99999	0.00807 1.00000
820	1.00000 0.00026	0.99947 0.92210	0.52223 0.99919	0.07093 0.99999	0.00732 1.00000
	1.00000 0.00026	0.99974 0.92127	0.55975 0.99918	0.07873 0.99999	0.00817 1.00000
830	1.00000 0.00024	0.99952 0.92119	0.52652 0.99918	0.07176 0.99999	0.00741 1.00000
	1.00000 0.00024	0.99976 0.92035	0.56413 0.99917	0.07965 0.99999	0.00827 1.00000
840	1.00000 0.00022	0.99956 0.92028	0.53078 0.99917	0.07260 0.99999	0.00750 1.00000
	1.00000 0.00022	0.99978 0.91943	0.56847 0.99916	0.08057 0.99999	0.00836 1.00000
850	1.00000 0.00019	0.99960 0.91936	0.53500 0.99916	0.07343 0.99999	0.00759 1.00000
	1.00000 0.00019	0.99981 0.91851	0.57277 0.99915	0.08149 0.99999	0.00846 1.00000
860	1.00000 0.00018	0.99963 0.91845	0.53918 0.99915	0.07427 0.99999	0.00768 1.00000
	1.00000 0.00018	0.99982 0.91759	0.57702 0.99914	0.08241 0.99999	0.00856 1.00000
870	1.00000 0.00016	0.99967 0.91755	0.54332 0.99914	0.07510 0.99999	0.00777 1.00000
	1.00000 0.00016	0.99984 0.91667	0.58123 0.99913	0.08333 0.99999	0.00866 1.00000
880	1.00000 0.00014	0.99970 0.91664	0.54742 0.99913	0.07593 0.99999	0.00785 1.00000
	1.00000 0.00014	0.99986 0.91576	0.58540 0.99912	0.08424 0.99999	0.00876 1.00000
890	1.00000 0.00013	0.99972 0.91573	0.55149 0.99912	0.07677 0.99999	0.00794 1.00000
	1.00000 0.00013	0.99987 0.91484	0.58953 0.99911	0.08516 0.99999	0.00886 1.00000
900	1.00000 0.00012	0.99975 0.91482	0.55552 0.99911	0.07760 0.99999	0.00803 1.00000
	1.00000 0.00012	0.99988 0.91393	0.59361 0.99910	0.08607 0.99999	0.00896 1.00000
910	1.00000 0.00011	0.99977 0.91392	0.55952 0.99910	0.07843 0.99999	0.00812 1.00000
	1.00000 0.00011	0.99989 0.91301	0.59766 0.99909	0.08699 0.99999	0.00906 1.00000
920	1.00000 0.00010	0.99979 0.91301	0.56348 0.99909	0.07926 0.99999	0.00821 1.00000
	1.00000 0.00010	0.99990 0.91210	0.60166 0.99908	0.08790 0.99999	0.00916 1.00000
930	1.00000 0.00009	0.99981 0.91211	0.56740 0.99908	0.08008 0.99999	0.00830 1.00000
	1.00000 0.00009	0.99991 0.91119	0.60563 0.99907	0.08881 0.99999	0.00926 1.00000
940	1.00000 0.00008	0.99982 0.91121	0.57129 0.99907	0.08091 0.99999	0.00839 1.00000
	1.00000 0.00008	0.99992 0.91028	0.60956 0.99906	0.08972 0.99999	0.00936 1.00000
950	1.00000 0.00007	0.99984 0.91031	0.57514 0.99906	0.08174 0.99999	0.00848 1.00000
	1.00000 0.00007	0.99993 0.90937	0.61344 0.99905	0.09063 0.99999	0.00946 1.00000
960	1.00000 0.00006	0.99985 0.90940	0.57896 0.99905	0.08257 0.99999	0.00857 1.00000
	1.00000 0.00006	0.99994 0.90846	0.61729 0.99904	0.09154 0.99999	0.00955 1.00000
970	1.00000 0.00006	0.99987 0.90850	0.58274 0.99904	0.08339 0.99999	0.00866 1.00000
	1.00000 0.00006	0.99994 0.90755	0.62110 0.99903	0.09245 0.99999	0.00965 1.00000
980	1.00000 0.00005	0.99988 0.90761	0.58649 0.99903	0.08422 0.99999	0.00875 1.00000
	1.00000 0.00005	0.99995 0.90664	0.62487 0.99902	0.09336 0.99999	0.00975 1.00000
990	1.00000 0.00005	0.99989 0.90671	0.59021 0.99902	0.08504 0.99999	0.00884 1.00000
	1.00000 0.00005	0.99995 0.90574	0.62861 0.99901	0.09426 0.99999	0.00985 1.00000
1000	1.00000 0.00004	0.99990 0.90581	0.59389 0.99901	0.08586 0.99999	0.00893 1.00000
	1.00000 0.00004	0.99996 0.90483	0.63230 0.99900	0.09517 0.99999	0.00995 1.00000
1100	1.00000 0.00002	0.99996 0.89688	0.62895 0.99891	0.09406 0.99999	0.00982 1.00000
	1.00000 0.00002	0.99998 0.89583	0.66731 0.99890	0.10417 0.99999	0.01094 1.00000
1200	1.00000 0.00001	0.99998 0.88805	0.66098 0.99881	0.10218 0.99999	0.01071 1.00000
	1.00000 0.00001	0.99999 0.88692	0.69899 0.99880	0.11308 0.99999	0.01193 1.00000
1300	1.00000 0.00000	0.99999 0.87930	0.69025 0.99872	0.11023 0.99999	0.01160 1.00000
	1.00000 0.00000	1.00000 0.87809	0.72765 0.99870	0.12191 0.99999	0.01292 1.00000
1400	1.00000 0.00000	1.00000 0.87063	0.71699 0.99862	0.11820 0.99999	0.01249 1.00000
	1.00000 0.00000	1.00000 0.86935	0.75358 0.99860	0.13065 0.99999	0.01390 1.00000
1500	1.00000 0.00000	1.00000 0.86205	0.74142 0.99852	0.12610 0.99999	0.01338 1.00000
	1.00000 0.00000	1.00000 0.86070	0.77704 0.99850	0.13930 0.99999	0.01489 1.00000
1600	1.00000 0.00000	1.00000 0.85356	0.76375 0.99842	0.13394 0.99998	0.01426 1.00000
	1.00000 0.00000	1.00000 0.85214	0.79827 0.99840	0.14786 0.99998	0.01587 1.00000
1700	1.00000 0.00000	1.00000 0.84515	0.78414 0.99832	0.14170 0.99998	0.01515 1.00000
	1.00000 0.00000	1.00000 0.84366	0.81747 0.99830	0.15634 0.99998	0.01686 1.00000
1800	1.00000 0.00000	1.00000 0.83682	0.80278 0.99822	0.14939 0.99998	0.01604 1.00000
	1.00000 0.00000	1.00000 0.83526	0.83485 0.99820	0.16474 0.99998	0.01784 1.00000
1900	1.00000 0.00000	1.00000 0.82857	0.81980 0.99812	0.15702 0.99998	0.01692 1.00000
	1.00000 0.00000	1.00000 0.82695	0.85057 0.99810	0.17305 0.99998	0.01882 1.00000
2000	1.00000 0.00000	1.00000 0.82041	0.83536 0.99802	0.16457 0.99998	0.01781 1.00000
	1.00000 0.00000	1.00000 0.81872	0.86480 0.99800	0.18128 0.99998	0.01980 1.00000
2100	1.00000 0.00000	1.00000 0.81233	0.84957 0.99793	0.17206 0.99998	0.01869 1.00000
	1.00000 0.00000	1.00000 0.81058	0.87767 0.99790	0.18942 0.99998	0.02078 1.00000
2200	1.00000 0.00000	1.00000 0.80432	0.86256 0.99783	0.17948 0.99998	0.01957 1.00000
	1.00000 0.00000	1.00000 0.80251	0.88932 0.99780	0.19749 0.99998	0.02176 1.00000

2300	1.00000 0.00000	1.00000 0.79640	0.87442 0.99773	0.18684 0.99998	0.02045 1.00000
	1.00000 0.00000	1.00000 0.79452	0.89986 0.99770	0.20548 0.99998	0.02274 1.00000
2400	1.00000 0.00000	1.00000 0.78855	0.88526 0.99763	0.19413 0.99998	0.02134 1.00000
	1.00000 0.00000	1.00000 0.78662	0.90939 0.99760	0.21338 0.99998	0.02371 1.00000
2500	1.00000 0.00000	1.00000 0.78078	0.89517 0.99753	0.20135 0.99998	0.02222 1.00000
	1.00000 0.00000	1.00000 0.77879	0.91802 0.99750	0.22121 0.99998	0.02469 1.00000
2600	1.00000 0.00000	1.00000 0.77309	0.90422 0.99743	0.20851 0.99997	0.02310 1.00000
	1.00000 0.00000	1.00000 0.77104	0.92582 0.99740	0.22896 0.99997	0.02567 1.00000
2700	1.00000 0.00000	1.00000 0.76547	0.91249 0.99733	0.21560 0.99997	0.02397 1.00000
	1.00000 0.00000	1.00000 0.76337	0.93289 0.99730	0.23663 0.99997	0.02664 1.00000
2800	1.00000 0.00000	1.00000 0.75793	0.92004 0.99723	0.22263 0.99997	0.02485 1.00000
	1.00000 0.00000	1.00000 0.75577	0.93928 0.99720	0.24423 0.99997	0.02761 1.00000
2900	1.00000 0.00000	1.00000 0.75046	0.92695 0.99714	0.22960 0.99997	0.02573 1.00000
	1.00000 0.00000	1.00000 0.74825	0.94506 0.99710	0.25175 0.99997	0.02858 1.00000
3000	1.00000 0.00000	1.00000 0.74306	0.93325 0.99704	0.23651 0.99997	0.02661 1.00000
	1.00000 0.00000	1.00000 0.74081	0.95029 0.99700	0.25919 0.99997	0.02955 1.00000
3100	1.00000 0.00000	1.00000 0.73574	0.93901 0.99694	0.24335 0.99997	0.02748 1.00000
	1.00000 0.00000	1.00000 0.73344	0.95502 0.99690	0.26656 0.99997	0.03052 1.00000
3200	1.00000 0.00000	1.00000 0.72849	0.94428 0.99684	0.25013 0.99997	0.02836 1.00000
	1.00000 0.00000	1.00000 0.72614	0.95930 0.99681	0.27386 0.99997	0.03149 1.00000
3300	1.00000 0.00000	1.00000 0.72131	0.94909 0.99674	0.25685 0.99997	0.02923 1.00000
	1.00000 0.00000	1.00000 0.71891	0.96318 0.99671	0.28109 0.99997	0.03246 1.00000
3400	1.00000 0.00000	1.00000 0.71420	0.95348 0.99664	0.26351 0.99997	0.03010 1.00000
	1.00000 0.00000	1.00000 0.71176	0.96668 0.99661	0.28824 0.99997	0.03343 1.00000
3500	1.00000 0.00000	1.00000 0.70717	0.95750 0.99654	0.27011 0.99997	0.03098 1.00000
	1.00000 0.00000	1.00000 0.70468	0.96986 0.99651	0.29532 0.99997	0.03439 1.00000
3600	1.00000 0.00000	1.00000 0.70020	0.96117 0.99644	0.27666 0.99996	0.03185 1.00000
	1.00000 0.00000	1.00000 0.69766	0.97273 0.99641	0.30234 0.99996	0.03536 1.00000
3700	1.00000 0.00000	1.00000 0.69330	0.96452 0.99635	0.28314 0.99996	0.03272 1.00000
	1.00000 0.00000	1.00000 0.69072	0.97532 0.99631	0.30928 0.99996	0.03632 1.00000
3800	1.00000 0.00000	1.00000 0.68647	0.96758 0.99625	0.28957 0.99996	0.03359 1.00000
	1.00000 0.00000	1.00000 0.68385	0.97767 0.99621	0.31615 0.99996	0.03729 1.00000
3900	1.00000 0.00000	1.00000 0.67970	0.97038 0.99615	0.29593 0.99996	0.03446 1.00000
	1.00000 0.00000	1.00000 0.67704	0.97980 0.99611	0.32296 0.99996	0.03825 1.00000
4000	1.00000 0.00000	1.00000 0.67301	0.97294 0.99605	0.30224 0.99996	0.03533 1.00000
	1.00000 0.00000	1.00000 0.67031	0.98172 0.99601	0.32969 0.99996	0.03921 1.00000
4100	1.00000 0.00000	1.00000 0.66638	0.97528 0.99595	0.30850 0.99996	0.03620 1.00000
	1.00000 0.00000	1.00000 0.66364	0.98346 0.99591	0.33636 0.99996	0.04017 1.00000
4200	1.00000 0.00000	1.00000 0.65981	0.97741 0.99585	0.31470 0.99996	0.03706 1.00000
	1.00000 0.00000	1.00000 0.65703	0.98504 0.99581	0.34297 0.99996	0.04113 1.00000
4300	1.00000 0.00000	1.00000 0.65331	0.97936 0.99575	0.32084 0.99996	0.03793 1.00000
	1.00000 0.00000	1.00000 0.65050	0.98646 0.99571	0.34950 0.99996	0.04209 1.00000
4400	1.00000 0.00000	1.00000 0.64687	0.98114 0.99566	0.32693 0.99996	0.03879 1.00000
	1.00000 0.00000	1.00000 0.64402	0.98775 0.99561	0.35598 0.99996	0.04305 1.00000
4500	1.00000 0.00000	1.00000 0.64050	0.98277 0.99556	0.33296 0.99996	0.03966 1.00000
	1.00000 0.00000	1.00000 0.63761	0.98892 0.99551	0.36239 0.99996	0.04400 1.00000
4600	1.00000 0.00000	1.00000 0.63418	0.98426 0.99546	0.33894 0.99995	0.04052 1.00000
	1.00000 0.00000	1.00000 0.63127	0.98997 0.99541	0.36873 0.99995	0.04496 1.00000
4700	1.00000 0.00000	1.00000 0.62794	0.98562 0.99536	0.34486 0.99995	0.04139 1.00000
	1.00000 0.00000	1.00000 0.62499	0.99093 0.99531	0.37501 0.99995	0.04591 1.00000
4800	1.00000 0.00000	1.00000 0.62175	0.98686 0.99526	0.35074 0.99995	0.04225 1.00000
	1.00000 0.00000	1.00000 0.61877	0.99179 0.99521	0.38123 0.99995	0.04687 1.00000
4900	1.00000 0.00000	1.00000 0.61562	0.98799 0.99516	0.35655 0.99995	0.04311 1.00000
	1.00000 0.00000	1.00000 0.61261	0.99257 0.99511	0.38739 0.99995	0.04782 1.00000
5000	1.00000 0.00000	1.00000 0.60956	0.98903 0.99506	0.36232 0.99995	0.04397 1.00000
	1.00000 0.00000	1.00000 0.60652	0.99328 0.99501	0.39348 0.99995	0.04877 1.00000
5100	1.00000 0.00000	1.00000 0.60355	0.98998 0.99497	0.36804 0.99995	0.04483 1.00000
	1.00000 0.00000	1.00000 0.60048	0.99392 0.99491	0.39952 0.99995	0.04972 1.00000
5200	1.00000 0.00000	1.00000 0.59760	0.99084 0.99487	0.37370 0.99995	0.04569 1.00000
	1.00000 0.00000	1.00000 0.59451	0.99450 0.99481	0.40549 0.99995	0.05067 1.00000
5300	1.00000 0.00000	1.00000 0.59171	0.99163 0.99477	0.37932 0.99995	0.04655 1.00000
	1.00000 0.00000	1.00000 0.58859	0.99502 0.99471	0.41141 0.99995	0.05162 1.00000
5400	1.00000 0.00000	1.00000 0.58588	0.99235 0.99467	0.38488 0.99995	0.04741 1.00000
	1.00000 0.00000	1.00000 0.58273	0.99550 0.99461	0.41727 0.99995	0.05257 1.00000

5500	1.00000 0.00000	1.00000 0.58011	0.99301 0.99457	0.39039 0.99995	0.04826 1.00000
	1.00000 0.00000	1.00000 0.57693	0.99592 0.99452	0.42307 0.99995	0.05352 1.00000
5600	1.00000 0.00000	1.00000 0.57439	0.99362 0.99447	0.39586 0.99994	0.04912 1.00000
	1.00000 0.00000	1.00000 0.57119	0.99631 0.99442	0.42881 0.99994	0.05446 1.00000
5700	1.00000 0.00000	1.00000 0.56873	0.99417 0.99437	0.40127 0.99994	0.04998 1.00000
	1.00000 0.00000	1.00000 0.56551	0.99666 0.99432	0.43449 0.99994	0.05541 1.00000
5800	1.00000 0.00000	1.00000 0.56313	0.99467 0.99428	0.40664 0.99994	0.05083 1.00000
	1.00000 0.00000	1.00000 0.55988	0.99698 0.99422	0.44012 0.99994	0.05635 1.00000
5900	1.00000 0.00000	1.00000 0.55758	0.99513 0.99418	0.41196 0.99994	0.05168 1.00000
	1.00000 0.00000	1.00000 0.55431	0.99727 0.99412	0.44569 0.99994	0.05729 1.00000
6000	1.00000 0.00000	1.00000 0.55209	0.99555 0.99408	0.41723 0.99994	0.05254 1.00000
	1.00000 0.00000	1.00000 0.54880	0.99753 0.99402	0.45120 0.99994	0.05824 1.00000
6100	1.00000 0.00000	1.00000 0.54665	0.99594 0.99398	0.42245 0.99994	0.05339 1.00000
	1.00000 0.00000	1.00000 0.54333	0.99776 0.99392	0.45667 0.99994	0.05918 1.00000
6200	1.00000 0.00000	1.00000 0.54126	0.99629 0.99388	0.42763 0.99994	0.05424 1.00000
	1.00000 0.00000	1.00000 0.53793	0.99798 0.99382	0.46207 0.99994	0.06012 1.00000
6300	1.00000 0.00000	1.00000 0.53593	0.99661 0.99378	0.43276 0.99994	0.05509 1.00000
	1.00000 0.00000	1.00000 0.53258	0.99817 0.99372	0.46742 0.99994	0.06106 1.00000
6400	1.00000 0.00000	1.00000 0.53065	0.99690 0.99369	0.43784 0.99994	0.05594 1.00000
	1.00000 0.00000	1.00000 0.52728	0.99834 0.99362	0.47272 0.99994	0.06200 1.00000
6500	1.00000 0.00000	1.00000 0.52542	0.99717 0.99359	0.44288 0.99994	0.05679 1.00000
	1.00000 0.00000	1.00000 0.52203	0.99850 0.99352	0.47797 0.99994	0.06293 1.00000
6600	1.00000 0.00000	1.00000 0.52024	0.99741 0.99349	0.44787 0.99993	0.05764 1.00000
	1.00000 0.00000	1.00000 0.51683	0.99864 0.99342	0.48317 0.99993	0.06387 1.00000
6700	1.00000 0.00000	1.00000 0.51511	0.99764 0.99339	0.45282 0.99993	0.05849 1.00000
	1.00000 0.00000	1.00000 0.51169	0.99877 0.99332	0.48831 0.99993	0.06481 1.00000
6800	1.00000 0.00000	1.00000 0.51004	0.99784 0.99329	0.45773 0.99993	0.05933 1.00000
	1.00000 0.00000	1.00000 0.50660	0.99889 0.99322	0.49340 0.99993	0.06574 1.00000
6900	1.00000 0.00000	1.00000 0.50501	0.99803 0.99319	0.46259 0.99993	0.06018 1.00000
	1.00000 0.00000	1.00000 0.50156	0.99900 0.99312	0.49844 0.99993	0.06667 1.00000
7000	1.00000 0.00000	1.00000 0.50004	0.99820 0.99310	0.46741 0.99993	0.06103 1.00000
	1.00000 0.00000	1.00000 0.49657	0.99909 0.99302	0.50343 0.99993	0.06761 1.00000
7100	1.00000 0.00000	1.00000 0.49511	0.99835 0.99300	0.47218 0.99993	0.06187 1.00000
	1.00000 0.00000	1.00000 0.49163	0.99918 0.99293	0.50837 0.99993	0.06854 1.00000
7200	1.00000 0.00000	1.00000 0.49023	0.99849 0.99290	0.47691 0.99993	0.06272 1.00000
	1.00000 0.00000	1.00000 0.48673	0.99926 0.99283	0.51327 0.99993	0.06947 1.00000
7300	1.00000 0.00000	1.00000 0.48540	0.99862 0.99280	0.48160 0.99993	0.06356 1.00000
	1.00000 0.00000	1.00000 0.48189	0.99933 0.99273	0.51811 0.99993	0.07040 1.00000
7400	1.00000 0.00000	1.00000 0.48062	0.99874 0.99270	0.48625 0.99993	0.06440 1.00000
	1.00000 0.00000	1.00000 0.47710	0.99939 0.99263	0.52290 0.99993	0.07133 1.00000
7500	1.00000 0.00000	1.00000 0.47588	0.99885 0.99260	0.49085 0.99993	0.06524 1.00000
	1.00000 0.00000	1.00000 0.47235	0.99945 0.99253	0.52765 0.99993	0.07226 1.00000
7600	1.00000 0.00000	1.00000 0.47119	0.99895 0.99251	0.49541 0.99992	0.06608 1.00000
	1.00000 0.00000	1.00000 0.46765	0.99950 0.99243	0.53235 0.99992	0.07318 1.00000
7700	1.00000 0.00000	1.00000 0.46655	0.99904 0.99241	0.49994 0.99992	0.06692 1.00000
	1.00000 0.00000	1.00000 0.46300	0.99955 0.99233	0.53700 0.99992	0.07411 1.00000
7800	1.00000 0.00000	1.00000 0.46195	0.99912 0.99231	0.50442 0.99992	0.06776 1.00000
	1.00000 0.00000	1.00000 0.45839	0.99959 0.99223	0.54161 0.99992	0.07504 1.00000
7900	1.00000 0.00000	1.00000 0.45740	0.99920 0.99221	0.50886 0.99992	0.06860 1.00000
	1.00000 0.00000	1.00000 0.45383	0.99963 0.99213	0.54617 0.99992	0.07596 1.00000
8000	1.00000 0.00000	1.00000 0.45289	0.99927 0.99211	0.51326 0.99992	0.06944 1.00000
	1.00000 0.00000	1.00000 0.44931	0.99967 0.99203	0.55069 0.99992	0.07688 1.00000
8100	1.00000 0.00000	1.00000 0.44843	0.99933 0.99201	0.51763 0.99992	0.07028 1.00000
	1.00000 0.00000	1.00000 0.44484	0.99970 0.99193	0.55516 0.99992	0.07781 1.00000
8200	1.00000 0.00000	1.00000 0.44401	0.99939 0.99192	0.52195 0.99992	0.07111 1.00000
	1.00000 0.00000	1.00000 0.44041	0.99973 0.99183	0.55959 0.99992	0.07873 1.00000
8300	1.00000 0.00000	1.00000 0.43964	0.99944 0.99182	0.52623 0.99992	0.07195 1.00000
	1.00000 0.00000	1.00000 0.43603	0.99975 0.99173	0.56397 0.99992	0.07965 1.00000
8400	1.00000 0.00000	1.00000 0.43531	0.99949 0.99172	0.53048 0.99992	0.07278 1.00000
	1.00000 0.00000	1.00000 0.43169	0.99978 0.99164	0.56831 0.99992	0.08057 1.00000
8500	1.00000 0.00000	1.00000 0.43102	0.99953 0.99162	0.53469 0.99992	0.07362 1.00000
	1.00000 0.00000	1.00000 0.42740	0.99980 0.99154	0.57260 0.99992	0.08149 1.00000
8600	1.00000 0.00000	1.00000 0.42677	0.99957 0.99152	0.53886 0.99991	0.07445 1.00000
	1.00000 0.00000	1.00000 0.42314	0.99982 0.99144	0.57686 0.99991	0.08241 1.00000

8700	1.00000 0.00000	1.00000 0.42256	0.99961 0.99143	0.54299 0.99991	0.07528 1.00000
	1.00000 0.00000	1.00000 0.41893	0.99983 0.99134	0.58107 0.99991	0.08332 1.00000
8800	1.00000 0.00000	1.00000 0.41840	0.99964 0.99133	0.54709 0.99991	0.07612 1.00000
	1.00000 0.00000	1.00000 0.41476	0.99985 0.99124	0.58524 0.99991	0.08424 1.00000
8900	1.00000 0.00000	1.00000 0.41428	0.99968 0.99123	0.55115 0.99991	0.07695 1.00000
	1.00000 0.00000	1.00000 0.41064	0.99986 0.99114	0.58936 0.99991	0.08515 1.00000
9000	1.00000 0.00000	1.00000 0.41020	0.99970 0.99113	0.55517 0.99991	0.07778 1.00000
	1.00000 0.00000	1.00000 0.40655	0.99988 0.99104	0.59345 0.99991	0.08607 1.00000
9100	1.00000 0.00000	1.00000 0.40615	0.99973 0.99103	0.55916 0.99991	0.07861 1.00000
	1.00000 0.00000	1.00000 0.40251	0.99989 0.99094	0.59749 0.99991	0.08698 1.00000
9200	1.00000 0.00000	1.00000 0.40215	0.99975 0.99094	0.56311 0.99991	0.07944 1.00000
	1.00000 0.00000	1.00000 0.39850	0.99990 0.99084	0.60150 0.99991	0.08790 1.00000
9300	1.00000 0.00000	1.00000 0.39819	0.99977 0.99084	0.56703 0.99991	0.08026 1.00000
	1.00000 0.00000	1.00000 0.39454	0.99991 0.99074	0.60546 0.99991	0.08881 1.00000
9400	1.00000 0.00000	1.00000 0.39426	0.99979 0.99074	0.57091 0.99991	0.08109 1.00000
	1.00000 0.00000	1.00000 0.39061	0.99992 0.99064	0.60939 0.99991	0.08972 1.00000
9500	1.00000 0.00000	1.00000 0.39038	0.99981 0.99064	0.57475 0.99991	0.08192 1.00000
	1.00000 0.00000	1.00000 0.38672	0.99993 0.99054	0.61328 0.99991	0.09063 1.00000
9600	1.00000 0.00000	1.00000 0.38653	0.99983 0.99054	0.57857 0.99990	0.08274 1.00000
	1.00000 0.00000	1.00000 0.38287	0.99993 0.99045	0.61713 0.99990	0.09154 1.00000
9700	1.00000 0.00000	1.00000 0.38272	0.99984 0.99044	0.58234 0.99990	0.08357 1.00000
	1.00000 0.00000	1.00000 0.37906	0.99994 0.99035	0.62094 0.99990	0.09244 1.00000
9800	1.00000 0.00000	1.00000 0.37895	0.99986 0.99035	0.58609 0.99990	0.08439 1.00000
	1.00000 0.00000	1.00000 0.37529	0.99994 0.99025	0.62471 0.99990	0.09335 1.00000
9900	1.00000 0.00000	1.00000 0.37522	0.99987 0.99025	0.58980 0.99990	0.08522 1.00000
	1.00000 0.00000	1.00000 0.37156	0.99995 0.99015	0.62844 0.99990	0.09426 1.00000
10000	1.00000 0.00000	1.00000 0.37152	0.99988 0.99015	0.59347 0.99990	0.08604 1.00000
	1.00000 0.00000	1.00000 0.36786	0.99995 0.99005	0.63214 0.99990	0.09516 1.00000
11000	1.00000 0.00000	1.00000 0.33650	0.99995 0.98917	0.62848 0.99989	0.09423 1.00000
	1.00000 0.00000	1.00000 0.33285	0.99998 0.98906	0.66715 0.99989	0.10417 1.00000
12000	1.00000 0.00000	1.00000 0.30477	0.99998 0.98819	0.66046 0.99988	0.10235 1.00000
	1.00000 0.00000	1.00000 0.30118	0.99999 0.98807	0.69882 0.99988	0.11308 1.00000
13000	1.00000 0.00000	1.00000 0.27604	0.99999 0.98721	0.68970 0.99987	0.11039 1.00000
	1.00000 0.00000	1.00000 0.27251	1.00000 0.98708	0.72749 0.99987	0.12191 1.00000
14000	1.00000 0.00000	1.00000 0.25001	1.00000 0.98624	0.71642 0.99986	0.11836 1.00000
	1.00000 0.00000	1.00000 0.24658	1.00000 0.98610	0.75342 0.99986	0.13064 1.00000
15000	1.00000 0.00000	1.00000 0.22644	1.00000 0.98526	0.74083 0.99985	0.12626 1.00000
	1.00000 0.00000	1.00000 0.22311	1.00000 0.98511	0.77689 0.99985	0.13929 1.00000
16000	1.00000 0.00000	1.00000 0.20509	1.00000 0.98429	0.76315 0.99984	0.13409 1.00000
	1.00000 0.00000	1.00000 0.20188	1.00000 0.98413	0.79812 0.99984	0.14786 1.00000
17000	1.00000 0.00000	1.00000 0.18576	1.00000 0.98331	0.78354 0.99983	0.14185 1.00000
	1.00000 0.00000	1.00000 0.18267	1.00000 0.98314	0.81733 0.99983	0.15634 1.00000
18000	1.00000 0.00000	1.00000 0.16824	1.00000 0.98234	0.80218 0.99982	0.14954 1.00000
	1.00000 0.00000	1.00000 0.16528	1.00000 0.98216	0.83472 0.99982	0.16473 1.00000
19000	1.00000 0.00000	1.00000 0.15238	1.00000 0.98137	0.81921 0.99981	0.15716 1.00000
	1.00000 0.00000	1.00000 0.14955	1.00000 0.98118	0.85045 0.99981	0.17304 1.00000
20000	1.00000 0.00000	1.00000 0.13802	1.00000 0.98040	0.83478 0.99980	0.16471 1.00000
	1.00000 0.00000	1.00000 0.13532	1.00000 0.98020	0.86468 0.99980	0.18127 1.00000
21000	1.00000 0.00000	1.00000 0.12500	1.00000 0.97943	0.84900 0.99979	0.17219 1.00000
	1.00000 0.00000	1.00000 0.12244	1.00000 0.97922	0.87756 0.99979	0.18942 1.00000
22000	1.00000 0.00000	1.00000 0.11322	1.00000 0.97846	0.86201 0.99978	0.17961 1.00000
	1.00000 0.00000	1.00000 0.11079	1.00000 0.97824	0.88921 0.99978	0.19748 1.00000
23000	1.00000 0.00000	1.00000 0.10254	1.00000 0.97749	0.87389 0.99977	0.18696 1.00000
	1.00000 0.00000	1.00000 0.10025	1.00000 0.97726	0.89975 0.99977	0.20547 1.00000
24000	1.00000 0.00000	1.00000 0.09288	1.00000 0.97652	0.88475 0.99976	0.19425 1.00000
	1.00000 0.00000	1.00000 0.09071	1.00000 0.97629	0.90929 0.99976	0.21337 1.00000
25000	1.00000 0.00000	1.00000 0.08412	1.00000 0.97556	0.89467 0.99975	0.20146 1.00000
	1.00000 0.00000	1.00000 0.08207	1.00000 0.97531	0.91793 0.99975	0.22120 1.00000
26000	1.00000 0.00000	1.00000 0.07619	1.00000 0.97459	0.90374 0.99974	0.20862 1.00000
	1.00000 0.00000	1.00000 0.07426	1.00000 0.97434	0.92574 0.99974	0.22895 1.00000
27000	1.00000 0.00000	1.00000 0.06901	1.00000 0.97363	0.91203 0.99973	0.21571 1.00000
	1.00000 0.00000	1.00000 0.06720	1.00000 0.97336	0.93280 0.99973	0.23662 1.00000
28000	1.00000 0.00000	1.00000 0.06250	1.00000 0.97266	0.91960 0.99972	0.22274 1.00000
	1.00000 0.00000	1.00000 0.06080	1.00000 0.97239	0.93920 0.99972	0.24422 1.00000

29000	1.00000 0.00000	1.00000 0.05661	1.00000 0.97170	0.92652 0.99971	0.22970 1.00000
	1.00000 0.00000	1.00000 0.05502	1.00000 0.97142	0.94498 0.99971	0.25174 1.00000
30000	1.00000 0.00000	1.00000 0.05127	1.00000 0.97074	0.93285 0.99970	0.23660 1.00000
	1.00000 0.00000	1.00000 0.04978	1.00000 0.97045	0.95022 0.99970	0.25918 1.00000
31000	1.00000 0.00000	1.00000 0.04644	1.00000 0.96978	0.93863 0.99969	0.24344 1.00000
	1.00000 0.00000	1.00000 0.04504	1.00000 0.96948	0.95496 0.99969	0.26655 1.00000
32000	1.00000 0.00000	1.00000 0.04206	1.00000 0.96882	0.94392 0.99968	0.25022 1.00000
	1.00000 0.00000	1.00000 0.04076	1.00000 0.96851	0.95924 0.99968	0.27385 1.00000
33000	1.00000 0.00000	1.00000 0.03809	1.00000 0.96786	0.94875 0.99967	0.25694 1.00000
	1.00000 0.00000	1.00000 0.03688	1.00000 0.96754	0.96312 0.99967	0.28108 1.00000
34000	1.00000 0.00000	1.00000 0.03450	1.00000 0.96690	0.95316 0.99966	0.26360 1.00000
	1.00000 0.00000	1.00000 0.03337	1.00000 0.96657	0.96663 0.99966	0.28823 1.00000
35000	1.00000 0.00000	1.00000 0.03125	1.00000 0.96595	0.95719 0.99965	0.27020 1.00000
	1.00000 0.00000	1.00000 0.03019	1.00000 0.96561	0.96981 0.99965	0.29531 1.00000
36000	1.00000 0.00000	1.00000 0.02830	1.00000 0.96499	0.96088 0.99964	0.27674 1.00000
	1.00000 0.00000	1.00000 0.02732	1.00000 0.96464	0.97268 0.99964	0.30232 1.00000
37000	1.00000 0.00000	1.00000 0.02563	1.00000 0.96403	0.96425 0.99963	0.28322 1.00000
	1.00000 0.00000	1.00000 0.02472	1.00000 0.96368	0.97528 0.99963	0.30927 1.00000
38000	1.00000 0.00000	1.00000 0.02322	1.00000 0.96308	0.96732 0.99962	0.28964 1.00000
	1.00000 0.00000	1.00000 0.02237	1.00000 0.96271	0.97763 0.99962	0.31614 1.00000
39000	1.00000 0.00000	1.00000 0.02103	1.00000 0.96213	0.97014 0.99961	0.29600 1.00000
	1.00000 0.00000	1.00000 0.02024	1.00000 0.96175	0.97976 0.99961	0.32294 1.00000
40000	1.00000 0.00000	1.00000 0.01905	1.00000 0.96118	0.97271 0.99960	0.30231 1.00000
	1.00000 0.00000	1.00000 0.01831	1.00000 0.96079	0.98169 0.99960	0.32968 1.00000
41000	1.00000 0.00000	1.00000 0.01725	1.00000 0.96022	0.97506 0.99959	0.30856 1.00000
	1.00000 0.00000	1.00000 0.01657	1.00000 0.95983	0.98343 0.99959	0.33635 1.00000
42000	1.00000 0.00000	1.00000 0.01562	1.00000 0.95927	0.97721 0.99958	0.31476 1.00000
	1.00000 0.00000	1.00000 0.01499	1.00000 0.95887	0.98501 0.99958	0.34295 1.00000
43000	1.00000 0.00000	1.00000 0.01415	1.00000 0.95833	0.97917 0.99957	0.32090 1.00000
	1.00000 0.00000	1.00000 0.01357	1.00000 0.95791	0.98643 0.99957	0.34949 1.00000
44000	1.00000 0.00000	1.00000 0.01282	1.00000 0.95738	0.98096 0.99956	0.32698 1.00000
	1.00000 0.00000	1.00000 0.01227	1.00000 0.95695	0.98773 0.99956	0.35596 1.00000
45000	1.00000 0.00000	1.00000 0.01161	1.00000 0.95643	0.98260 0.99955	0.33301 1.00000
	1.00000 0.00000	1.00000 0.01111	1.00000 0.95600	0.98889 0.99955	0.36237 1.00000
46000	1.00000 0.00000	1.00000 0.01051	1.00000 0.95548	0.98410 0.99954	0.33899 1.00000
	1.00000 0.00000	1.00000 0.01005	1.00000 0.95504	0.98995 0.99954	0.36872 1.00000
47000	1.00000 0.00000	1.00000 0.00952	1.00000 0.95454	0.98547 0.99953	0.34491 1.00000
	1.00000 0.00000	1.00000 0.00909	1.00000 0.95409	0.99091 0.99953	0.37500 1.00000
48000	1.00000 0.00000	1.00000 0.00863	1.00000 0.95359	0.98672 0.99952	0.35078 1.00000
	1.00000 0.00000	1.00000 0.00823	1.00000 0.95313	0.99177 0.99952	0.38122 1.00000
49000	1.00000 0.00000	1.00000 0.00781	1.00000 0.95265	0.98786 0.99952	0.35660 1.00000
	1.00000 0.00000	1.00000 0.00744	1.00000 0.95218	0.99256 0.99951	0.38738 1.00000
50000	1.00000 0.00000	1.00000 0.00708	1.00000 0.95171	0.98891 0.99951	0.36236 1.00000
	1.00000 0.00000	1.00000 0.00674	1.00000 0.95123	0.99326 0.99950	0.39347 1.00000
51000	1.00000 0.00000	1.00000 0.00641	1.00000 0.95077	0.98986 0.99950	0.36808 0.99999
	1.00000 0.00000	1.00000 0.00610	1.00000 0.95028	0.99390 0.99949	0.39951 0.99999
52000	1.00000 0.00000	1.00000 0.00580	1.00000 0.94982	0.99074 0.99949	0.37374 0.99999
	1.00000 0.00000	1.00000 0.00552	1.00000 0.94933	0.99448 0.99948	0.40548 0.99999
53000	1.00000 0.00000	1.00000 0.00526	1.00000 0.94888	0.99153 0.99948	0.37935 0.99999
	1.00000 0.00000	1.00000 0.00499	1.00000 0.94838	0.99501 0.99947	0.41140 0.99999
54000	1.00000 0.00000	1.00000 0.00476	1.00000 0.94795	0.99226 0.99947	0.38491 0.99999
	1.00000 0.00000	1.00000 0.00452	1.00000 0.94743	0.99548 0.99946	0.41725 0.99999
55000	1.00000 0.00000	1.00000 0.00431	1.00000 0.94701	0.99293 0.99946	0.39042 0.99999
	1.00000 0.00000	1.00000 0.00409	1.00000 0.94649	0.99591 0.99945	0.42305 0.99999
56000	1.00000 0.00000	1.00000 0.00391	1.00000 0.94607	0.99354 0.99945	0.39588 0.99999
	1.00000 0.00000	1.00000 0.00370	1.00000 0.94554	0.99630 0.99944	0.42879 0.99999
57000	1.00000 0.00000	1.00000 0.00354	1.00000 0.94513	0.99409 0.99944	0.40130 0.99999
	1.00000 0.00000	1.00000 0.00335	1.00000 0.94459	0.99665 0.99943	0.43448 0.99999
58000	1.00000 0.00000	1.00000 0.00320	1.00000 0.94420	0.99460 0.99943	0.40666 0.99999
	1.00000 0.00000	1.00000 0.00303	1.00000 0.94365	0.99697 0.99942	0.44010 0.99999
59000	1.00000 0.00000	1.00000 0.00290	1.00000 0.94326	0.99507 0.99942	0.41198 0.99999
	1.00000 0.00000	1.00000 0.00274	1.00000 0.94271	0.99726 0.99941	0.44567 0.99999
60000	1.00000 0.00000	1.00000 0.00263	1.00000 0.94233	0.99549 0.99941	0.41725 0.99999
	1.00000 0.00000	1.00000 0.00248	1.00000 0.94176	0.99752 0.99940	0.45119 0.99999

61000	1.00000 0.00000 1.00000 0.00000	1.00000 0.00238 1.00000 0.00224	1.00000 0.94140 1.00000 0.94082	0.99588 0.99940 0.99776 0.99939	0.42247 0.99999 0.45665 0.99999	
62000	1.00000 0.00000 1.00000 0.00000	1.00000 0.00216 1.00000 0.00203	1.00000 0.94047 1.00000 0.93988	0.99623 0.99939 0.99797 0.99938	0.42764 0.99999 0.46206 0.99999	
63000	1.00000 0.00000 1.00000 0.00000	1.00000 0.00195 1.00000 0.00184	1.00000 0.93954 1.00000 0.93894	0.99656 0.99938 0.99816 0.99937	0.43277 0.99999 0.46741 0.99999	
64000	1.00000 0.00000 1.00000 0.00000	1.00000 0.00177 1.00000 0.00166	1.00000 0.93861 1.00000 0.93800	0.99686 0.99937 0.99834 0.99936	0.43785 0.99999 0.47271 0.99999	
65000	1.00000 0.00000 1.00000 0.00000	1.00000 0.00160 1.00000 0.00150	1.00000 0.93768 1.00000 0.93707	0.99713 0.99936 0.99850 0.99935	0.44289 0.99999 0.47796 0.99999	
66000	1.00000 0.00000 1.00000 0.00000	1.00000 0.00145 1.00000 0.00136	1.00000 0.93675 1.00000 0.93613	0.99737 0.99935 0.99864 0.99934	0.44788 0.99999 0.48315 0.99999	
67000	1.00000 0.00000 1.00000 0.00000	1.00000 0.00131 1.00000 0.00123	1.00000 0.93582 1.00000 0.93520	0.99760 0.99934 0.99877 0.99933	0.45283 0.99999 0.48829 0.99999	
68000	1.00000 0.00000 1.00000 0.00000	1.00000 0.00119 1.00000 0.00111	1.00000 0.93490 1.00000 0.93426	0.99781 0.99933 0.99889 0.99932	0.45773 0.99999 0.49338 0.99999	
69000	1.00000 0.00000 1.00000 0.00000	1.00000 0.00108 1.00000 0.00101	1.00000 0.93397 1.00000 0.93333	0.99800 0.99932 0.99899 0.99931	0.46259 0.99999 0.49843 0.99999	
70000	1.00000 0.00000 1.00000 0.00000	1.00000 0.00098 1.00000 0.00091	1.00000 0.93305 1.00000 0.93239	0.99817 0.99931 0.99909 0.99930	0.46741 0.99999 0.50342 0.99999	
71000	1.00000 0.00000 1.00000 0.00000	1.00000 0.00088 1.00000 0.00082	1.00000 0.93213 1.00000 0.93146	0.99833 0.99930 0.99918 0.99929	0.47218 0.99999 0.50836 0.99999	
72000	1.00000 0.00000 1.00000 0.00000	1.00000 0.00080 1.00000 0.00075	1.00000 0.93120 1.00000 0.93053	0.99847 0.99929 0.99925 0.99928	0.47691 0.99999 0.51325 0.99999	
73000	1.00000 0.00000 1.00000 0.00000	1.00000 0.00073 1.00000 0.00068	1.00000 0.93028 1.00000 0.92960	0.99860 0.99928 0.99932 0.99927	0.48159 0.99999 0.51809 0.99999	
74000	1.00000 0.00000 1.00000 0.00000	1.00000 0.00066 1.00000 0.00061	1.00000 0.92936 1.00000 0.92867	0.99872 0.99927 0.99939 0.99926	0.48624 0.99999 0.52289 0.99999	
75000	1.00000 0.00000 1.00000 0.00000	1.00000 0.00060 1.00000 0.00055	1.00000 0.92844 1.00000 0.92774	0.99883 0.99926 0.99945 0.99925	0.49084 0.99999 0.52764 0.99999	
76000	1.00000 0.00000 1.00000 0.00000	1.00000 0.00054 1.00000 0.00050	1.00000 0.92752 1.00000 0.92682	0.99893 0.99925 0.99950 0.99924	0.49540 0.99999 0.53234 0.99999	
77000	1.00000 0.00000 1.00000 0.00000	1.00000 0.00049 1.00000 0.00045	1.00000 0.92660 1.00000 0.92589	0.99902 0.99924 0.99955 0.99923	0.49993 0.99999 0.53699 0.99999	
78000	1.00000 0.00000 1.00000 0.00000	1.00000 0.00044 1.00000 0.00041	1.00000 0.92569 1.00000 0.92496	0.99911 0.99923 0.99959 0.99922	0.50441 0.99999 0.54160 0.99999	
79000	1.00000 0.00000 1.00000 0.00000	1.00000 0.00040 1.00000 0.00037	1.00000 0.92477 1.00000 0.92404	0.99919 0.99922 0.99963 0.99921	0.50885 0.99999 0.54616 0.99999	
80000	1.00000 0.00000 1.00000 0.00000	1.00000 0.00036 1.00000 0.00034	1.00000 0.92386 1.00000 0.92312	0.99926 0.99921 0.99966 0.99920	0.51325 0.99999 0.55067 0.99999	
81000	1.00000 0.00000 1.00000 0.00000	1.00000 0.00033 1.00000 0.00030	1.00000 0.92294 1.00000 0.92219	0.99932 0.99920 0.99970 0.99919	0.51761 0.99999 0.55514 0.99999	
82000	1.00000 0.00000 1.00000 0.00000	1.00000 0.00030 1.00000 0.00027	1.00000 0.92203 1.00000 0.92127	0.99938 0.99919 0.99973 0.99918	0.52193 0.99999 0.55957 0.99999	
83000	1.00000 0.00000 1.00000 0.00000	1.00000 0.00027 1.00000 0.00025	1.00000 0.92112 1.00000 0.92035	0.99943 0.99918 0.99975 0.99917	0.52622 0.99999 0.56395 0.99999	
84000	1.00000 0.00000 1.00000 0.00000	1.00000 0.00024 1.00000 0.00022	1.00000 0.92021 1.00000 0.91943	0.99948 0.99917 0.99978 0.99916	0.53046 0.99999 0.56829 0.99999	
85000	1.00000 0.00000 1.00000 0.00000	1.00000 0.00022 1.00000 0.00020	1.00000 0.91929 1.00000 0.91851	0.99953 0.99916 0.99980 0.99915	0.53467 0.99999 0.57259 0.99999	
86000	1.00000 0.00000 1.00000 0.00000	1.00000 0.00020 1.00000 0.00018	1.00000 0.91839 1.00000 0.91759	0.99957 0.99915 0.99982 0.99914	0.53884 0.99999 0.57684 0.99999	
87000	1.00000 0.00000 1.00000 0.00000	1.00000 0.00018 1.00000 0.00017	1.00000 0.91748 1.00000 0.91668	0.99960 0.99914 0.99983 0.99913	0.54297 0.99999 0.58105 0.99999	
88000	1.00000 0.00000 1.00000 0.00000	1.00000 0.00016 1.00000 0.00015	1.00000 0.91657 1.00000 0.91576	0.99964 0.99913 0.99985 0.99912	0.54706 0.99999 0.58522 0.99999	
89000	1.00000 0.00000 1.00000 0.00000	1.00000 0.00015 1.00000 0.00014	1.00000 0.91566 1.00000 0.91485	0.99967 0.99912 0.99986 0.99911	0.55112 0.99999 0.58935 0.99999	
90000	1.00000 0.00000 1.00000 0.00000	1.00000 0.00013 1.00000 0.00012	1.00000 0.91476 1.00000 0.91393	0.99970 0.99911 0.99988 0.99910	0.55514 0.99999 0.59343 0.99999	
91000	1.00000 0.00000 1.00000 0.00000	1.00000 0.00012 1.00000 0.00011	1.00000 0.91385 1.00000 0.91302	0.99972 0.99910 0.99989 0.99909	0.55913 0.99999 0.59748 0.99999	
92000	1.00000 0.00000 1.00000 0.00000	1.00000 0.00011 1.00000 0.00010	1.00000 0.91295 1.00000 0.91211	0.99975 0.99909 0.99990 0.99908	0.56308 0.99999 0.60148 0.99999	

93000	1.00000	0.00000	1.00000	0.00010	1.00000	0.91204	0.99977	0.99908	0.56700	0.99999
	1.00000	0.00000	1.00000	0.00009	1.00000	0.91119	0.99991	0.99907	0.60545	0.99999
94000	1.00000	0.00000	1.00000	0.00009	1.00000	0.91114	0.99979	0.99907	0.57087	0.99999
	1.00000	0.00000	1.00000	0.00008	1.00000	0.91028	0.99992	0.99906	0.60937	0.99999
95000	1.00000	0.00000	1.00000	0.00008	1.00000	0.91024	0.99981	0.99906	0.57472	0.99999
	1.00000	0.00000	1.00000	0.00007	1.00000	0.90937	0.99993	0.99905	0.61326	0.99999
96000	1.00000	0.00000	1.00000	0.00007	1.00000	0.90934	0.99982	0.99905	0.57853	0.99999
	1.00000	0.00000	1.00000	0.00007	1.00000	0.90846	0.99993	0.99904	0.61711	0.99999
97000	1.00000	0.00000	1.00000	0.00007	1.00000	0.90844	0.99984	0.99904	0.58231	0.99999
	1.00000	0.00000	1.00000	0.00006	1.00000	0.90756	0.99994	0.99903	0.62092	0.99999
98000	1.00000	0.00000	1.00000	0.00006	1.00000	0.90754	0.99985	0.99903	0.58605	0.99999
	1.00000	0.00000	1.00000	0.00006	1.00000	0.90665	0.99994	0.99902	0.62469	0.99999
99000	1.00000	0.00000	1.00000	0.00006	1.00000	0.90664	0.99987	0.99902	0.58976	0.99999
	1.00000	0.00000	1.00000	0.00005	1.00000	0.90574	0.99995	0.99901	0.62843	0.99999
100000	1.00000	0.00000	1.00000	0.00005	1.00000	0.90574	0.99988	0.99901	0.59343	0.99999
	1.00000	0.00000	1.00000	0.00005	1.00000	0.90484	0.99995	0.99900	0.63212	0.99999

3.2 DC-network

As substantiated in section 3, we will be mainly concerned with the data link layer implementing anonymous medium access and the upper sublayer of the physical layer implementing superposed sending by exchanging or generating key bits as well as superposing them and possibly a message in a synchronized fashion.

As noted in section 1.4.1, someone can identify a sender, if and only if he knows all keys the sender shares. David Chaum describes in [Cha3_85] how someone violating the anonymous medium access protocol can be traced in the long run:

Whenever a violation of the medium access protocol is noted, all user stations have to publish the corresponding bits of all keys they used. If the violator does not lie (or lies self-contradicting), he is identified.

If he lies intelligently, there will be disputes with at least one other user station concerning a key bit. By removing the corresponding key from the key graph (cf. section 1.4.1), a violator will get more and more isolated. The same is true for teams of violators.

Obviously, one should try to restrict this tracing to protocol information (for which a message was not yet sent) or dummy messages, to avoid tracing both violators and legitimate senders of messages.

Of course, we could use this protocol to diagnose permanent and even transient faults as well, but would probably need a second communication system to diagnose the first. Please note that we have multiple communication systems with respect to most faults (and all faults of the sublayer implementing superposed sending), if we use different XOR-gates, amplifiers and timing circuitry for each multiplexed channel, as mentioned in section 2.2.2.1.

Since faults do not behave intelligently, we can reduce the expense to diagnose them. The expense of the algorithm described above grows at least proportionally with the number of user stations, whereas the expense of our specialized algorithm (described later) only grows with the logarithm of the number of user stations.
Whenever faulty behaviour of the DC-network is detected (by medium access protocol violations or error detecting codes safeguarding messages which could not suffer from a collision) and lasts for a while (suggesting that there is a permanent fault), all user stations try to reinitialize the medium access protocol (to tolerate transient errors of the physical layer which may have disturbed the medium access protocol) and, if this is not successful, switch from the sender anonymity mode to the fault diagnosis mode. To diagnose and recover permanent faults of the upper sublayer of the physical layer, they execute the following protocol:

Each station initiates self-diagnosis: after establishing a recovery point [AnLe_81] of the states of its key generators, it loads random seeds in pairs into its pseudorandom key generators and superposes their output with a random message.
If the result is not the random message, the station is faulty and broadcasts this. All other stations discard keys shared with that station and reenter (one fault assumption) the sender anonymity mode.
If the result is the random message, the station uses the recovery point to establish its former state.
At that point of the protocol, there are three probable faults:

1) a station is so faulty, that it cannot diagnose itself and broadcast the outcome, or
2) the synchronization of key generation and superposition has been lost between (at least) two stations sharing a key, or
3) there is a fault in the communication system.

To distinguish between these types of faults and to locate them, the number of superposed keys is halved successively, and say 100 (new and never reused) bits are superposed corresponding to a key graph with only half as many vertices.
If the outcome is 100 zeros, a fault is in the other half and the communication system contains no fault (with a probability of $1 - 2^{-100}$, if a stuck at zero fault at the last XOR-gate has been excluded before this test, which can e.g. be done by letting all stations send 100 random bits, which should yield a non-zero result with the same probability). If the outcome is not 100 zeros, at least one fault concerning the generation, synchronization or superposition of these keys is present or the communication system is faulty, and we go on halving.
When the key graph has only one vertex left and the outcome of the superposition is not 100 zeros, both stations exchange new keys (to recover from key synchronization faults) and try again.
If the outcome is again not 100 zeros, both discard this key and exchange a key with a different third and fourth station. Both new pairs try again, one after the other.
If both tries fail, there is probably a fault in the communication system (which should be diagnosable with logarithmic expense as well).
If one try fails, the corresponding station is assumed to be faulty and taken out of operation. This is broadcasted to all stations, which discard keys shared with that station and reenter (one fault assumption) the sender anonymity mode.

Of course, manifold variations of this fault diagnosis and recovery protocol are possible and appropriate after the probability of single and multiple faults is known. For example, before reentering sender anonymity mode, halves, which would not

be tested using the above protocol, should be tested, given that the probability of multiple faults is significant.
So, not the details of fault diagnosis and recovery are important, but that they are possible using logarithmic time expense and without any compromise of unobservability, due to the separation of sender anonymity mode and fault diagnosis mode (bits used in sender anonymity mode are not used for fault diagnosis and cryptographically strong pseudorandom bit generators [BlMi_84, VaVa_85] prevent an attacker from discovering relationships between bits of the same pseudorandom sequence, which may be used in both modes).

3.3 RING-network

As substantiated in section 3, we will be mainly concerned with the data link layer implementing anonymous ring access, the physical layer implementing digital signal (re)generation and the medium implementing signal propagation.

Transient faults of the physical layer disturbing the anonymous ring access protocol can be tolerated by reinitializing the ring access protocol with the techniques which are common use for token rings and slotted rings.

If the digital signal (re)generation or the medium suffers from a permanent fault, only reconfiguration of the ring (which must incorporate redundant links for this purpose) can help in both cases whereas the closing of a station bypass (bypass attached to each station which physically closes the ring if the station fails or is powered off) can help only in the first case. Since the first is the more powerful concept and has strong interactions with unobservability whereas the second has no interactions with unobservability if only the station itself can close its bypass and since both concepts may be combined easily, we will only consider the more powerful but more critical (with respect to unobservability) one.
To avoid single point observability of user stations, the

concept of ring wiring concentrators [BCKK_83, KeMM_83] must not be used and the user stations themselves must be able to bypass each other.

To be able to tolerate both permanent link and user station faults, the RING-network can be implemented by a braided ring (figure 14) and special protocols [Mann_85]. The key ideas are:

As long as no permanent fault occurs, the links connecting adjacent stations may be operated as one RING-network, the other links as a second independent one for odd numbers of stations, using the protocols described in section 1.4.2, 2.1.1 and 2.2.1. All remarks concerning unobservability and performance made there remain valid and we have doubled transmission capacity.

When a station (say, number 5) observes permanent severe signal degradation or permanent signal outage at one of its input lines, it broadcasts this. As long as the permanent fault cannot be repaired, our station under consideration ignores input on that line. Fortunately, one RING-network remains operational as explained below.

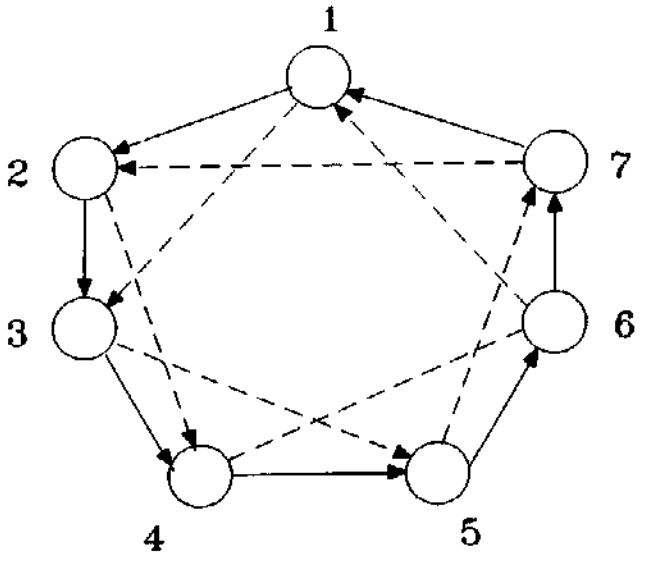
If the sender of that line (say, number 4) does not broadcast on its other line that it is all right, the fault is supposed to be due to a station fault and the corresponding station is circumvented as depicted in figure 14.

If the sender of that line broadcasts on its other line that it is all right, its predecessor (station number 3) has to transmit all traffic on both its lines.

The two recipients (stations 4 and 5) transmit complementary halves of the slots, as depicted in figure 14. This has the advantage that the station assembling the halves (station 6) cannot observe whether one of its predecessors sends. Stations 4 and 5 can receive with the whole bandwidth of the ring, but only use half its bandwidth to send, until our faulty link is repaired.

The anonymity is that of two rings with half the bandwidth, one through station 4 and the other through station 5. The fact that on both of them an additional station can receive does not matter since the receiver does not change anything in our normal

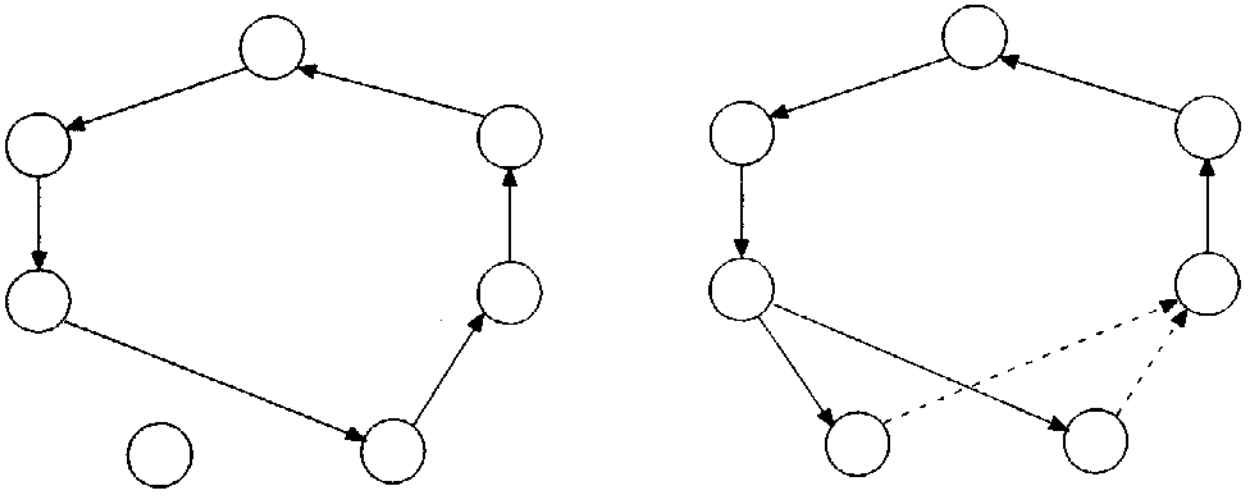
all stations and lines up:



first ring second ring

station 4 down:

line between station 4 and station 5 down:



complementary halves of the slots

Fig. 14: Fault tolerant RING-network on the physical structure of a braided ring

medium access protocols. If we use one slot to establish a duplex channel between another station and station 4 or 5, the latter can only send with a probability of 1/2. The situation that such a station is addressed by someone wishing to establish

a duplex channel in a slot which the station may not use for sending is an infrequent event but may undermine anonymity against the connection initiator.
This is only a special case of the general problem that an addressee may be identified if he cannot respond due to a station failure and his station is the only station which is down at the moment and the sender knows this.

Please note that there are possibilities where an attacker surrounding two stations in the RING-network in which no fault occurred surrounds one station in the reconfigured one. For example, an attacker controlling the stations 2 and 5 can pretend that the line between station 4 and station 5 is faulty and then observe whether station 3 sends.
But even in this case, the attacker must be very close to station 3 and could therefore use other means to observe its owner (cf. section 1.4.2).

A quantitative examination of the reliability improvement caused by this scheme is given in [Mann_85]. The results are very promising.

3.4 Hierarchical networks

Hierarchical anonymous networks can be made fault tolerant by incorporating in their (sub)networks the fault tolerance mechanisms described in the three preceding sections.
The connection between these (sub)networks can be made fault tolerant by the use of multiple gateways between each adjacent pair of (sub)networks.

Since the sending or receiving of gateways may be observable in the SBNS, arbitrary protocols between them may be used to manage the redundancy for fault tolerance.
Andreas Mann discusses this and other points for the SBNS (using RING-networks as broadcast subnetworks) in great depth in [Mann_85].

4 Who should establish which parts of the network and be responsible for the quality of service

The previous four sections dealt with mechanisms which, when incorporated in a communication network, allow users to send and to receive anonymously, as well as maintain high performance and reliability of the network.
On our way we noticed that these mechanisms are virtual concepts and therefore can be implemented in different layers, but that there is a canonical implementation (depicted in figure 5) promising highest performance and reliability at given cost.
In this section we have to think on a related question, namely, who should establish and maintain which parts of the network, and as a consequence, be responsible for the quality of service.
Of course, the manufacturer, establisher, and maintainer of the mechanisms which provide unobservability must be prevented from implanting Trojan Horses into the corresponding equipment.

Here, we are faced with the following dilemma:
If we let everybody use an arbitrary user station, as we tacitly assumed in most of the previous sections, consisting of equipment not concerned with communication (called DTE = Data Terminal Equipment by CCITT) and an arbitrary interface to the network (called DCE = Data Circuit-terminating Equipment by CCITT [Tane 81 p. 237, 238]), no organization will be able and willing to be responsible for the quality of service, i.e. performance and reliability, at least if faulty behavior or unsatisfactory performance of one user station affects the quality of service for other users, like in a RING- or DC-network.
On the other hand, if we let one organization manufacture, establish, and maintain all network equipment including all parts of the user stations, this organization will be able and willing to be responsible for the quality of service, but such an organization has too much power in its hands to be efficiently controlled by the users.
All we can do is to give an organization control over the DCEs, but not the DTEs and to make the DCE as small and therefore its

design as easy to check for Trojan Horses as possible. Then we can accept that one, or some few suppliers which mutually accept the quality of their equipment, provide all DCEs used.
These suppliers have to publish their designs as well as their design criteria. Consumer associations like the german "Stiftung Warentest" should look for Trojan Horses in these DCEs continuously, which takes only great efforts at the beginning when the design must be checked, but is simple when only the identical reproduction must be controlled.
DCEs will only comprise a few digital chips and analog sender(s) and receiver(s). Therefore, maintenance will be rare. Especially, no software can be modified by the supplier by remote maintenance over the communication network, as is usually the case for today's switching centers, whereby the maintainer can establish and remove Trojan Horses at will.

If MIXes are operated by users, the DCE comprises a switching center and a bulk of very powerful crypto equipment. Perhaps we should eliminate the switching function of some MIXes and group the MIXes operated by some users in a fixed order, in which every message has to pass all MIXes. But even the crypto equipment and its coordination equipment are not very easy to check for Trojan Horses and may need maintenance more often than desired.

For a DC-network, DCEs must be equipped with fast and secure pseudorandom bit generators and implement the medium access protocol as well as all mechanisms needed for fault tolerance (all this cannot be in the DTE because it affects the quality of service for others). For this case of a large ISDN with one network operator it is even more necessary than in some local area network that the pseudorandom bit generators should be provably secure: Otherwise the algorithm may contain a trapdoor which helps the provider to observe users, or may be broken some months after it has become a de facto standard for this network and cannot be changed at reasonable cost.
There are two further comments in order concerning provably secure pseudorandom bit generators:

1) So far, they are proved only under the assumption that a well known number theoretic problem is hard. Since nobody can prove the hardness of such a problem, there may be big surprises!
2) All pseudorandom bit generators 'proved' to date are very expensive to implement.

For a RING-network, DCEs must be equipped with fast digital signal (re)generation and transmission capability and implement the ring access protocol as well as all mechanisms needed for fault tolerance.
Compared with the two other DCEs mentioned above, this is a very simple DCE.
Since our proofs make no assumptions concerning computational complexity, there will be no surprises!

After we answered the question of our section title as far as ISDNs are concerned, we have to remark again that where neither performance nor reliability is of utmost concern, users may use their DTEs to implement a MIX-, DC- or even RING-network on an arbitrary network provided by anybody. Of course, the unobservability (as far as Trojan Horses are concerned) in such a network will be beyond doubt, but at the price of low cost-efficiency, low performance, low reliability and, perhaps, suspicion from the rest of the society what these strange people have to hide.

5 Concluding remarks on networks without user observability

The previous five sections dealt with the design, establishment, and operation of a network with high performance and reliability which allows its users to send and to receive anonymously.
As mentioned in section 0, the content of a message can be hidden by using end-to-end encryption.
If using the network is not for free, the charges must either be paid anonymously with each use of the network (e.g. by anonymous numbered accounts [Pfit_84, Pfi1_83, Bürk_86, BüPf_86] or digital banknotes [Cha4_85, Cha8_85, Bürk_86, BüPf_86]), which seems rather troublesome, or measured anonymously (e.g. by safeguarded counters at user stations [Pfit_84, Pfi1_83]), or paid by flat rates.

The initially mentioned services like electronic mail, ordering of newspapers or home banking can be implemented by higher protocols on top of such a network.
If identification is required instead of anonymity, well known authentication schemes can be used. Even concurrent identification is possible [Gold_83].
Otherwise it is necessary to implement the services in a way which preserves the anonymity of the network. This must be proved in addition to proofs that the implementation fulfils its normal specification, e.g. security against fraud [WaPf_85, PPW_86].

It should be mentioned that many communication services where users nowadays have to identify themselves can be used in an anonymous way in the future if there is a protocol that allows people to act under several pseudonyms and to transform documents that carry one of these pseudonyms into documents carrying another of their own pseudonyms, in a secure and anonymous way [Cha1_84, Cha2_85, Cha8_85].

An appropriate layering of these functions is shown in figure 15.

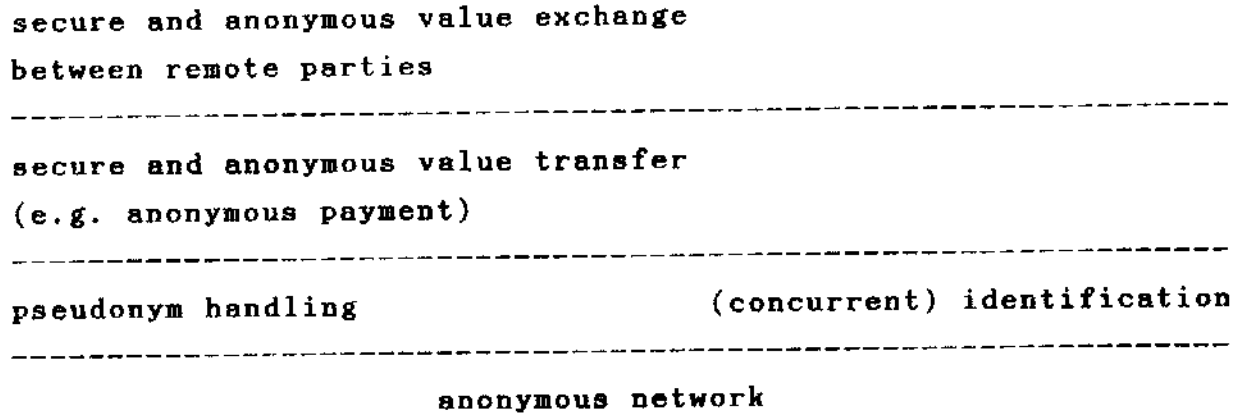


Fig. 15: Layering of higher protocols on top of an anonymous network

6 Application of the techniques to the confinement problem

The explained techniques to build anonymous networks can be applied to another problem as well. In [Lamp_73] Butler Lampson described the "confinement problem", the problem how a program can be confined not to leak information it receives to do its job to someone not explicitly allowed to receive that information.
In [RuRa_83] John Rushby and Brian Randell discuss a partial solution to the confinement problem in distributed systems. Their solution uses so called "Trustworthy Network Interface Units", abbreviated TNIUs, to enforce isolation where necessary. These TNIUs control all the traffic between the untrusted host computers over the untrusted local area network. Rushby and Randell note at page 60 of their article that a program trying to leak information may modulate the destination addresses of messages it sends. Since TNIUs do not encrypt addresses in Rusby and Randell's local area network, this can easily be interpreted by a wiretapper. The only countermeasure they propose is randomly addressed dummy traffic between the TNIUs, but this is only a partial solution: it makes the covert channel noisy.
This covert channel can be completely closed if broadcast is used on the local area network (this is usual anyway) and the TNIUs translate the addresses generated by the untrusted host computers into visible implicit addresses, which are only used once, or invisible implicit addresses (cf. section 1.2).

If the TNIUs use sender anonymity techniques, they close the covert channel of the frequence of sending messages partially without giving up the ability of dynamic bandwidth sharing. Static bandwidth sharing (each TNIU sends with a fixed rate irrespective how much it has to send) seems to be the price to close this covert channel completely.

Acknowledgements

Some survey parts of this report were written in cooperation with Michael Waidner and published in [PfWa_85]. Dr. Klaus Dittrich read and polished these parts very carefully.
Besides to them, I am grateful to David Chaum for sending his drafts to our research group as well as to Holger Bürk for programming the formulas to evaluate the reliability of various fault tolerant MIX schemes.
I thank them, Klaus Echtle, Andreas Mann and especially Birgit Pfitzmann for a lot of useful comments and stimulating discussions.

Literature

Agne_85 Gordon B. Agnew: Secrecy and Privacy in a Local Area Network Environment; Advances in Cryptology, Proceedings of EUROCRYPT 84, A Workshop on the Theory and Application of Cryptographic Techniques, April 9-11, 1984, Paris, France, Edited by T. Beth, N. Cot and I. Ingemarsson, Lecture Notes in Computer Science LNCS 209, Springer-Verlag Heidelberg, 1985, pp. 349..363

Alba_83 A. Albanese: Star Network With Collision-Avoidance Circuits; The Bell System Technical Journal (BSTJ) Vol. 62, Nu. 3, March 1983, pp. 631..638

AlFi_77 Marcelo Alonso, Edward J. Finn: Physik; Deutsche Übersetzung von Anneliese Schimpl, Herausgegeben von Wolfgang Muschik; Inter European Editions, Amsterdam, 1977

AlSc_83 Bowen Alpern, Fred B. Schneider: Key exchange Using 'Keyless Cryptography'; Information Processing Letters Vol. 16, 26 February 1983, pp. 79..81

AnLe_81 T. Anderson, P. A. Lee: Fault Tolerance - Principles and Practice; Prentice Hall, Englewood Cliffs, New Jersey, 1981

BaBr_85 E. E. Basch, T. G. Brown: Introduction to Coherent Optical Fiber Transmission; IEEE Communications Magazine Vol. 23, No. 5, May 1985, pp. 23..30

Bara_64 Paul Baran: On Distributed Communications: IX. Security, Secrecy, and Tamper-Free Considerations; Memorandum RM-3765-PR, Aug. 1964, The Rand Corporation, Santa Monica, California

BaSa_85 R. J. S. Bates, L. A. Sauer: Jitter accommodation in token-passing ring LANs; IBM Journal on Research and Development Vol. 29, No. 6, November 1985, pp. 580..587

BCKK_83 Werner Bux, Felix H. Closs, Karl Kuemmerle, Heinz J. Keller, Hans R. Mueller: Architecture and Design of a Reliable Token-Ring Network; IEEE Journal on Selected Areas in Communications Vol. SAC-1, No. 5, November 1983, pp. 756..765

BeEn_85 Larry A. Bergman, Sverre T. Eng: A Synchronous Fiber Optic Ring Local Area Network for Multigigabit/s Mixed-Traffic Communication; IEEE Journal on Selected Areas in Communications Vol. SAC-3, No. 6, November 1985, pp. 842..848

Blan_84 Robert P. Blanc: Local Area Network Standards; digest of papers compcon spring 84, Twenty-eighth IEEE Computer Society International Conference, San Francisco, California, February 27-March 1 1984, pp. 252..254

BlMi_84 Manuel Blum, Silvio Micali: How to Generate Cryptographically Strong Sequences of Pseudo-Random Bits; SIAM J. Comput. Vol. 13, No. 4, November 1984, pp. 850..864

BoLe_83 Peter I. P. Boulton, E. Stewart Lee: Bus, Ring, Star and Tree Local Area Networks; Computer Communication Review Vol. 13, No. 3, July 1983, pp. 19..24

Brau_83 Ewald Braun: BIGFON Brings the Optical Waveguide into the Subscriber Area; Optical Communications, A Telecommunications Review, SIEMENS, John Wiley & Sons Limited (Title of German original edition: telcom report Nachrichtenübertragung mit Licht), 1983, pp. 136..139

BüPf_86 Holger Bürk, Andreas Pfitzmann: Value transfer systems enabling security and unobservability; will be submitted to IFIP/Sec. '86, Fourth International Conference and Exhibition on Computer Security, Monte Carlo, December 1986

Bürk_86 Holger Bürk: Digitale Zahlungssysteme und betrugssicherer, anonymer Wertetransfer; Studienarbeit am Institut für Informatik IV, Universität Karlsruhe, April 1986

Bürl_84 Gabriele Bürle: Leistungsvergleich von Sternnetz und Schieberegister-Ringnetz; Studienarbeit, Univ. Karlsruhe, 1984

Bürl_85 Gabriele Bürle: Leistungsbewertung von Vermittlungs-/Verteilnetzen; Diplomarbeit, Univ. Karlsruhe, Mai 1985

Chau_81 David Chaum: Untraceable Electronic Mail, Return Addresses, and Digital Pseudonyms; CACM Vol. 24, Nu. 2, Feb. 1981, pp. 84..88

Chau_84 David Chaum: Design Concepts for Tamper Responding Systems; Advances in Cryptology, Proceedings of Crypto 83, A Workshop on the Theory and Application of Cryptographic Techniques, August 22-24, 1983, University of California, Santa Barbara, Edited by David Chaum, Plenum Press, New York, 1984, pp. 387..392

Cha1_84 David Chaum: A New Paradigm for Individuals in the Information Age; Proc. of the 1984 Symp. on Security and

Privacy, IEEE, Apr. 1984, Oakland, California, pp. 99..103

Cha2_85 David Chaum: Showing Credentials Without Identification. Signatures Transferred Between Unconditionally Unlinkable Pseudonyms; Eurocrypt 85, Draft, received May 13, 1985;

Cha3_85 David Chaum: The Dining Cryptographers Problem. Unconditional Sender Anonymity; Draft, received May 13, 1985;

Cha4_85 David Chaum: Privacy Protected Payments. Unconditional Payer and/or Payee Anonymity; Draft, received May 13, 1985;

Cha8_85 David Chaum: Security Without Identification: Transaction Systems to Make Big Brother Obsolete; CACM Vol. 28, Nu. 10, Oct. 1985, pp. 1030..1044

ClLe_81 Felix Closs, Robert P. Lee: A Multi-Star Broadcast Network for Local-Area Communication; Local Networks for Computer Communications, IFIP, International Workshop on Local Networks; Anthony West, Philippe Janson (eds.), Zürich, August 1980, North-Holland Publishing Company, 1981, pp. 61..80

DaPr_84 D. W. Davies, W. L. Price: Security for Computer Networks, An Introduction to Data Security in Teleprocessing and Electronic Funds Transfer; John Wiley & Sons, Chichester, New York, 1984

DaZi_83 John D. Day, Hubert Zimmermann: The OSI Reference Model; Proceedings of the IEEE Vol. 71, Nu. 12, December 1983, pp. 1334..1340

Denn_82 Dorothy E. Denning: Cryptography and Data Security; Addison-Wesley Publishing Company, Reading, Mass.; 1982

DiHe_76 Whitfield Diffie, Martin E. Hellman: New Directions in Cryptography; IEEE Transactions on Information Theory, Vol. IT-22, No. 6, November 1976, pp. 644..654

DiHe_79 Whitfield Diffie, Martin E. Hellman: Privacy and Authentication: An Introduction to Cryptography; Proceedings of the IEEE, Vol. 67, No. 3, March 1979, pp. 397..427

Eck_85 Wim van Eck: Electromagnetic Radiation from Video Display Units: An Eavesdropping Risk? Computers & Security Vol. 4, Nu. 4, December 1985, pp. 269..286

ECMA89_85 ECMA European Computer Manufacturers Association: Standard ECMA-89 Local Area Networks Token Ring Technique; 2 nd Edition - March 1985; nearly identical with: ANSI/IEEE Std 802.5-1985, ISO/DP 8802/5 "Local Area Networks, Token Ring Access Method", Approved December 13, 1984 IEEE Standards Board, Approved March 19, 1985 American National Standards Institute

FaLa_75 David J. Farber, Kenneth C. Larson: Network Security Via Dynamic Process Renaming; Fourth Data Communications Symp., Oct. 1975, Quebec City, Canada, pp. 8-13..8-18

Finl_84 Marion R. Finley: Optical Fibers in Local Area Networks; IEEE Communications Magazine Vol. 22, No. 8, August 1984, pp. 22..35

GDRB_85 Robert C. Goodfellow, B. T. Debney, Graham J. Rees, Jens Buus: Optoelectronic Components for Multigigabit Sytems; IEEE/OSA Journal of Lightwave Technology Vol. LT-3, No. 6, December 1985, pp. 1170..1179

Gilo_81 Wolfgang K. Giloi: Rechnerarchitektur; Springer-Verlag, Heidelberger Taschenbücher Sammlung Informatik 208, 1981

GöKü_85 Ernst-Heinrich Göldner, Paul J. Kühn: Integration of Voice and Data in the Local Area; First International Conference on Data Communications in the ISDN Era, Tel-Aviv University, March 4-5, 1985, North Holland, IFIP 1985, pp. 103..117

Göld_85 Ernst-Heinrich Göldner: An Integrated Circuit/Packet Switching Local Area Network - Performance Analysis and Comparison of Strategies; 11th International Teletraffic Congress (ITC), Kyoto, Japan, September 1985; reprinted in: Computer Networks and ISDN Systems Vol. 10, 1985, pp. 211..219

Gold_83 Oded Goldreich: On Concurrent Identification Protocols; Laboratory for Computer Science, Massachusetts Institute of Technology, MIT/LCS/TM-250, December 1983

HeFi_80 Clayton L. Henderson, Allan M. Fine: Motion, Intrusion and Tamper Detection for Surveillance and Containment; Sandia Laboratories, Albuquerque, New Mexico 87185; SAND79-0792, Printed March 1980

Höck_85 Gunter Höckel: Untersuchung der Datenschutzeigenschaften von Ringzugriffsmechanismen; Diplomarbeit, Univ. Karlsruhe, August 1985

HöPf_85 Gunter Höckel, Andreas Pfitzmann: Untersuchung der Datenschutzeigenschaften von Ringzugriffsmechanismen; 1. GI-Fachtagung "Datenschutz und Datensicherung", Okt. 1985, München, IFB Band 113, Springer-Verlag, Heidelberg, pp. 113..127

Horg_85 John Horgan: Thwarting the information thieves; IEEE Spectrum Vol. 22, Nu. 7, July 1985, pp. 30..41

Hor2_85 John Horgan: Inventor seeks to warn Government of threat from laser-based bug; The Institute, IEEE, October 1985, p. 8

Kanz_83 Jürgen Kanzow: BIGFON: Preparation for the Use of Optical Fiber Technology in the Local Network of the Deutsche Bundespost; IEEE Journal on Selected Areas in Communications Vol. SAC-1, No. 3, April 1983, pp. 436..439

Karg_77 Paul A. Karger: Non-Discretionary Access Control for Decentralized Computing Systems; Master Thesis, MIT, Laboratory for Computer Science, May 1977, Report MIT/LCS/TR-179

KeMM_83 Heinz J. Keller, Heinrich Meyr, Hans R. Müller: Transmission Design Criteria for a Synchronous Token Ring; IEEE Journal on Selected Areas in Communications

Vol. SAC-1, No. 5, November 1983, pp. 721..733

Kent_80 Stephen Thomas Kent: Protecting Externally Supplied Software in Small Computers; PhD-Thesis, Massachusetts Institute of Technology, Laboratory for Computer Science, 545 Technology Square, Cambridge, Massachusetts 02139, September 1980, Report MIT/LCS/TR-255

Kühn_85 P. J. Kühn: Diensteintegration für Lokal- und Weitverkehrsnetze; GI/NTG Tutorium und Tagung "Kommunikation in Verteilten Systemen - Anwendungen, Betrieb und Grundlagen -", 11.-15. März 1985, Tutoriumsband, D. Heger, G. Krüger, O. Spaniol, W. Zorn (eds.), pp. 41..63

KuSY_84 James F. Kurose, Mischa Schwartz, Yechiam Yemini: Multiple-Access Protocols and Time-Constrained Communication; acm computing surveys Vol. 16, Nu. 1, March 1984, pp. 43..70

Lamp_73 Butler W. Lampson: A Note on the Confinement Problem; Communications of the ACM Vol. 16, Nu. 10, October 1973, pp. 613..615

LeBo_83 E. Stewart Lee, Peter I. P. Boulton: The Principles and Performance of Hubnet: A 50 Mbit/s Glass Fiber Local Area Network; IEEE Journal on Selected Areas in Communications Vol. SAC-1, No. 5, November 1983, pp. 711..720

Mann_85 Andreas Mann: Fehlertoleranz und Datenschutz in Ringnetzen; Diplomarbeit, Univ. Karlsruhe, Okt. 1985

Mass_81 James L. Massey: Collision-Resolution Algorithms and Random-Access Communications; Multi-User Communication Systems; Edited by G. Longo; CISM Courses and Lectures No. 265, International Centre for Mechanical Sciences; Springer-Verlag Wien, New York, 1981, pp. 73..137

McLi_85 R. W. McLintock: Overview of International Standards for Transmission Impairments Affecting Digital Telecommunications Networks; Computer Networks and ISDN Systems Vol. 9, Nu. 5, May 1985, pp. 339..344

OSI_83 ISO: Basic reference model for Open Systems Interconnection; ISO 7498, 1983

Parn_74 David Parnas: On a "Buzzword": Hierarchical Structure; Information Processing 74, Proc. of IFIP Congress 74, Stockholm, Sweden, August 5-10, 1974, North-Holland Publishing Company, pp. 336..339

PfHä_82 Andreas Pfitzmann, Hermann Härtig: Grenzwerte der Zuverlässigkeit von Parallel-Serien-Systemen; Fehlertolerierende Rechnersysteme, GI-Fachtagung München, März 1982, Informatik-Fachberichte Band 54, Springer-Verlag, Berlin, Heidelberg, New York, 1982, pp. 1..16

Pfit_82 Andreas Pfitzmann: Konfigurierung und Modellierung von Mehrmikrorechnern aus um Zuverlässigkeitsanforderungen erweiterten ADA-Programmen; Interner Bericht Nr. 8/82, Institut für Informatik IV, Fakultät für Informatik, Universität Karlsruhe, Februar 1982

Pfit_83 Andreas Pfitzmann: Ein Vermittlungs-/Verteilnetz zur Erhöhung des Datenschutzes in Bildschirmtext-ähnlichen Neuen Medien; 13. Jahrestagung der GI, Okt. 1983, Univ. Hamburg, IFB Band 73, Springer-Verlag Heidelberg, pp. 411..418

Pfit_84 Andreas Pfitzmann: A switched/broadcast ISDN to decrease user observability; 1984 Intern. Zurich Seminar on Digital Communications, March 1984, Zurich, Switzerland, Swiss Federal Inst. of Tech., Proc. IEEE Cat. No. 84CH1998-4 pp. 183..190

Pfit_85 Andreas Pfitzmann: Technischer Datenschutz in diensteintegrierenden Digitalnetzen - Problemanalyse, Lösungsansätze und eine angepaßte Systemstruktur; 1. GI-Fachtagung "Datenschutz und Datensicherung", Okt. 1985, München, IFB Band 113, Springer-Verlag, Heidelberg, pp. 96..112

Pfi1_83 A. Pfitzmann: Ein dienstintegriertes digitales Vermittlungs-/Verteilnetz zur Erhöhung des Datenschutzes; Fak. f. Inform., Univ. Karlsruhe, Interner Bericht 18/83, Dez. 1983

PfPW_86 Andreas Pfitzmann, Birgit Pfitzmann, Michael Waidner: Technischer Datenschutz in diensteintegrierenden Digitalnetzen - Warum und wie? Revised and extended version of [Pfit_85], submitted to DuD, 'Datenschutz und Datensicherung, Informationsrecht, Kommunikationssysteme", Vieweg Verlag, Wiesbaden

PfWa_85 Andreas Pfitzmann, Michael Waidner: Networks without user observability -- design options; Eurocrypt 85, A Workshop on the Theory and Application of Cryptographic Techniques, April 9-11, 1985, Johannes-Kepler-University, Linz, Austria, IACR - International Association for Cryptologic Research, General Chairman: Franz Pichler, Program Chairman: Thomas Beth; Proceedings will be published by Springer-Verlag Heidelberg in the series LNCS; Revised and extended version entitled "Networks without user observability" will be submitted to Computers & Security, North-Holland

PoKl_78 G. J. Popek, C. S. Kline: Issues in Kernel Design; Operating Systems, An Advanced Course, Ed. by R. Bayer et. al.; LNCS 60, 1978; Springer-Verlag, Heidelberg, pp. 209..227

PPW_86 Andreas Pfitzmann, Birgit Pfitzmann, Michael Waidner: Rechtssicherheit trotz Anonymität in offenen digitalen Systemen; will be submitted to Computer und Recht (CuR), Verlag Dr. Otto Schmidt KG, Köln

Rayc_85 Dipankar Raychaudhuri: Announced Retransmission Random Access Protocols; IEEE Transactions on Communications Vol. COM-33, No. 11, November 1985, pp. 1183..1190

Röhr_82 Johannes Röhrich: Hierarchische Systeme; Informatik-Spektrum Band 5, Heft 2, Juli 1982, pp. 123..124

RuRa_83 John Rushby, Brian Randell: A Distributed Secure System; Computer, IEEE, Vol. 16, Nu. 7, July 1983, pp. 55..67

Schö_84 Helmut Schön: The Deutsche Bundespost on its Way towards the ISDN; Zeitschrift für das Post- und Fernmeldewesen Heft 6 vom 27. Juni 1984

ScSc_84 Christian Schwarz-Schilling (ed.): Konzept der Deutschen Bundespost zur Weiterentwicklung der Fernmeldeinfrastruktur; Herausgeber: Der Bundesminister für das Post- und Fernmeldewesen, Stab 202, Bonn, 1984

Sham_79 Adi Shamir: How to Share a Secret; Communications of the ACM Vol. 22, Nu. 11, November 1979, pp. 612..613

Shan_49 C. E. Shannon: Communication Theory of Secrecy Systems; Bell Syst. Tech. J., Vol. 28, No. 4, Oct. 1949, pp. 656..715

Shan_82 J. George Shanthikumar: Recursive Algorithm to Evaluate the Reliability of a Consecutive-k-out-of-n:F System; IEEE Transactions on Reliability Vol. R-31, No. 5, December 1982, pp. 442..443

Simm_85 Gustavus J. Simmons: How to (Selectively) Broadcast a Secret; Proceedings of the 1985 Symposium on Security and Privacy, April 22-24, 1985, Oakland, California, IEEE Computer Society, pp. 108..113

Stan_85 I. W. Stanley: A Tutorial Review of Techniques for Coherent Optical Fiber Transmission Systems; IEEE Communications Magazine Vol. 23, No. 8, August 1985, pp. 37..53

SuSY_84 Tatsuya Suda, Mischa Schwartz, Yechiam Yemini: Protocol Architecture of a Tree Network with Collision Avoidance Switches; Links for the Future; Science, Systems & Services for Communications, P. Dewilde and C. A. May (eds.); Proceedings of the International Conference on Communications - ICC '84, Amsterdam, The Netherlands, May 14-17, 1984, IEEE, Elsevier Science Publishers B. V. (North-Holland), pp. 423..427

Sze_85 Daniel T. W. Sze: A Metropolitan Area Network; IEEE Journal on Selected Areas in Communications Vol. SAC-3, No. 6, November 1985, pp. 815..824

Tane_81 Andrew S. Tanenbaum: Computer Networks; Prentice-Hall, Englewood Cliffs, N. J., 1981

Tasa_83 Shuji Tasaka: Stability and Performance of the R-ALOHA Packet Broadcast System; IEEE Transactions on Computers, Vol. C-32, No. 8, August 1983, pp. 717..726

Thom_84 Ken Thompson: Reflections on Trusting Trust; CACM, Vol. 27, No. 8, Aug. 1984, pp. 761..763

Toba_80 Fouad A. Tobagi: Multiaccess Protocols in Packet Communication Systems; IEEE Transactions on Communications, Vol. COM-28, No. 4, April 1980, pp. 468..488

Unge_84 Hans-Georg Unger: Trends in Optical Communications; Links for the Future; Science, Systems & Services for Communications, P. Dewilde and C. A. May (eds.); Proceedings of the International Conference on Communications - ICC '84, Amsterdam, The Netherlands, May 14-17, 1984, IEEE, Elsevier Science Publishers B. V. (North-Holland), pp. 153..158

VaVa_85 Umesh V. Vazirani, Vijay V. Vazirani: Efficient and Secure Pseudo-Random Number Generation (extended abstract); Advances in Cryptology, Proceedings of Crypto 84, A Workshop on the Theory and Application of Cryptographic Techniques, August 19-22, 1984, University of California, Santa Barbara, Edited by G. R. Blakley and David Chaum, Lecture Notes in Computer Science LNCS 196, Springer-Verlag Heidelberg, 1985, pp. 193..202

VoKe_83 Victor L. Voydock, Stephen T. Kent: Security Mechanisms in High-Level Network Protocols; acm computing surveys Vol. 15, No. 2, June 1983, pp. 135..171

VoKe_85 Victor L. Voydock, Stephen T. Kent: Security in High-level Network Protocols; IEEE Communications Magazine Vol. 23, Nu. 7, July 1985, pp. 12..24

WaGo_84 William M. Waite, Gerhard Goos: Compiler Construction; Texts and Monographs in Computer Science, Springer-Verlag Heidelberg, 1984

Waid_85 Michael Waidner: Datenschutz und Betrugssicherheit garantierende Kommunikationsnetze. Systematisierung der Datenschutzmaßnahmen und Ansätze zur Verifikation der Betrugssicherheit; Diplomarbeit, Fak. f. Inform., Univ. Karlsruhe, Interner Bericht 19/85, Aug. 1985

WaPf_85 Michael Waidner, Andreas Pfitzmann: Betrugssicherheit trotz Anonymität. Abrechnung und Geldtransfer in Netzen; 1. GI-Fachtagung "Datenschutz und Datensicherung", Okt. 1985, München, IFB Band 113, Springer-Verlag, Heidelberg, pp. 128..141; Revised version appears in DuD, "Datenschutz und Datensicherung, Informationsrecht, Kommunikationssysteme", Vieweg Verlag, Wiesbaden

Yao_82 Andrew C. Yao: Theory and Applications of Trapdoor Functions; 23rd Symposium on Foundations of Computer Science, November 3-5, 1982, pp. 80..91

Yemi_83 Yechiam Yemini: TINKERNET: Or, Is there Life between LANs and PBXs? Integrating Communication for World Progress; Proceedings of the International Conference on Communications - ICC '83, Boston, Massachusetts, June 19-22, 1983, IEEE, pp. 1501..1505