Real-Time Mixes: A Bandwidth-Efficient Anonymity Protocol

Anja Jerichow, Jan Müller, Andreas Pfitzmann, *Member, IEEE*, Birgit Pfitzmann, and Michael Waidner

***Abstract*—We present techniques for efficient anonymous communication with real-time constraints as necessary for services like telephony, where a continuous data stream has to be transmitted.**

For concreteness, we present the detailed protocols for the narrow-band ISDN (Integrated Services Digital Network), although the heart of our techniques—anonymous channels—can also be applied to other networks. For ISDN, we achieve the same data rate as without anonymity, using the same subscriber lines and without any significant modifications to the long-distance network. A precise performance analysis is given.

Our techniques are based on mixes, a method for anonymous communication for e-mail-like services introduced by D. Chaum.

***Index Terms*—Anonymity, ISDN, privacy, public-key cryptography, real-time communication, security.**

I. Introduction

ONE OF THE key questions of a global information infrastructure is privacy protection for its users. This includes not only the confidentiality of message contents, that can be implemented by encryption, but also mechanisms to hide address and routing information. Such privacy may be desired both with respect to outsiders and with respect to information providers the user accesses. Some examples are:

1) individual acts requiring special protection like contacting Alcoholics Anonymous;
2) the fact that the entire pattern of URL's a user accesses has quite a potential for abuse;
3) the protection of contacts between companies.

Note that wherever anonymity is not desired, identification can easily be added in higher layers (this is done anyway because network addresses are not trustworthy enough), whereas one cannot implement any anonymous service in higher layers (at least not at a reasonable cost) if the network addresses identify its users.

Several techniques are known that allow privacy *even if* an adversary may be tapping lines or have (legal or illegal) access to a switching center; see [1] for an overview. The best-known technique for anonymous communication, and the one we build upon, are mixes [2]. Although all basic techniques are quite old, real interest in applying them in practice has only come up recently. In [3], the basic mix protocol was used for anonymously submitting messages to the messaging service of a network of mobile computers. Several mix schemes for anonymous e-mail have actually been built, such as Mixmaster [4] and Babel [5]. However, mixes in their original form imply both a significant data expansion and significant delays, and therefore it was often considered infeasible to apply them to services with higher bandwidth and real-time requirements. In this paper, we present the first protocol that overcomes these problems.

To be concrete, we present our protocols in the context of *voice and data communication* on the narrowband ISDN (Integrated Services Digital Network) as standardized and actually built by most European PTT's (Post Telegraph Telephone companies). In these networks, each user (i.e., the user's NT (network terminating) device) is connected to a local exchange via an exclusive wire, whereas the bandwidth is shared in the long-distance network. Voice communication means strict real time during a call and a certain upper bound on the setup time. The ISDN requirements are that:

1) two bit-transparent duplex channels with 64 kb/s each must be offered on two given data channels of exactly this bandwidth;
2) an additional signaling channel of 16 kb/s is available;
3) any additional messages we need should fit into the signaling structure of the given ISDN.

Our techniques can be adapted to other networks as long as a certain delay at the start of a connection is tolerable. For instance, this was done for mobile communication based on the GSM standard in [6] and [7]. In [8], similar ideas were applied to synchronous communication over TCP/IP networks.

Mixes are not the only means for anonymous communication. DC networks [9], [10] can offer perfect sender anonymity, but can be implemented efficiently only on specific network topologies (e.g., ring). On general networks, the bandwidth expansion is proportional to the number of participants within an anonymity set. Other schemes achieve only a lower degree of anonymity, like Rings [11], [12] and Crowds [13].

In Section II, we briefly sketch the original mixes. In Section III, we present our protocols in an abstract way, i.e., in the language of the security community. Section IV contains a precise specification of the protocols for the real ISDN. The

Manuscript received March 4, 1997; revised September 25, 1997. An earlier version of this paper was presented at IFIP/Sec'91, "Kommunikation in verteilten Systemen 1991."

A. Jerichow, J. Müller, and A. Pfitzmann are with the Technische Universität Dresden, Theoretische Informatik, D-01062 Dresden, Germany (e-mail: jerichow@inf.tu-dresden.de; jm4@inf.tu-dresden.de; pfitza@inf.tu-dresden.de).

B. Pfitzmann is with Universität des Saarlandes, FB Informatik, D-66123 Saarbrücken, Germany (e-mail: pfitzmann@cs.uni-sb.de).

M. Waidner is with IBM Research Division, CH-8803 Rüschlikon, Switzerland (e-mail: wmi@zurich.ibm.com).

Publisher Item Identifier S 0733-8716(98)01103-2.

performance is evaluated in Section V. The security achieved is discussed in Section VI, which also shows that billing for network services works in spite of anonymity.

II. Basic Mixes

The basic idea of mixes [2] is that each message is sent over a series of independent stations, the mixes. Each mix collects a number of messages (called batch), discards repeats, changes the outlooks of the remaining messages by a cryptographic operation, and outputs them in a different order. Thus an outsider, even if he taps all the lines, cannot follow the passage of a message through several mixes unless all these mixes cooperate with him.

The cryptographic operations in the basic scheme [see (2.1)] are as follows:

$$\begin{aligned} N_{m+1} &:= N \\ N_i &:= c_i(A_{i+1}, r_{i+1}, N_{i+1}) \\ &\quad \text{for } i = m, m-1, \cdots, 1. \end{aligned} \tag{2.1}$$

The sender encrypts the message N plus some random bits r_i with a public key c_i of each mix M_i addressed by A_i, starting with the last one. This can be visualized as packing the message in successive envelopes. The outermost envelope can only be removed by the first mix, the next envelope by the second mix, etc. This corresponds to decryption with their private keys. The last mix forwards the intended message N to the recipient. (Of course, it may still be in an envelope, i.e., encrypted for the intended recipient; this is independent of the mix protocol.)

Even from this brief sketch, one can see that basic mixes are only suitable for nonreal-time communication.

1) There is a significant delay because a mix must wait until it has a sufficiently large amount of messages to mix.
2) Each mix must perform an asymmetric cryptographic operation on the message, which takes a certain time and can only start when at least a long block of the message has arrived.
3) Probabilistic encryption must be used: otherwise, the adversary could observe the messages output by a mix, reencrypt each of them with the public key of the mix, and simply compare which ciphertext equals which input message. Thus each ciphertext is longer by at least the number of random bits r_i used than the previous one, which adds up to a significant bandwidth expansion if several mixes are used.

We will only present protocol details for our modified mix schemes, but two more issues are useful to be kept in mind: First, the scheme sketched so far keeps the sender anonymous from the recipient, but a different scheme has to be used if the recipient is supposed to answer without being able to identify the sender. Secondly, no encryption system entirely hides the length of messages. Thus it does not make sense to mix messages whose ciphertexts can be distinguished simply by their lengths in the same batch.

III. Abstract Real-Time Mix Protocols

We present our protocols in two steps. By modifying the original mixes accordingly, first we introduce mix channels, which are a mechanism to mix continuous data streams. There will be two versions, one to keep a sender anonymous from a recipient and one the other way round. These mix channels will later serve as building blocks. However, in particular because of the varying length of phone calls, they cannot simply be used to mix entire connections. Thus, in the second step, we will build the entire abstract protocols by using mix channels for time slices (Real-Time Mixes). They are combined with other ingredients to guarantee sufficient size of the batches without additional cost and for the initial establishment of anonymous connections. We understand anonymous connection as follows: A is only anonymous among the people whose channels are mixed with hers by a local exchange (LE), in the following denoted by $LE(A)$. All subscribers of one LE build an *anonymity set* by equally acting to the LE. On the network side, it is not possible to map actions of the anonymity set to a single member. If A is member of the anonymity set of $LE(A)$ and B of $LE(B)$, for long distance calls one can only tell if anyone from $LE(A)$ is communicating with anyone from $LE(B)$.

A. Building Blocks: Mix Channels

We need an asymmetric and a symmetric encryption system [14]. We denote asymmetric encryption and decryption keys as c_x and d_x, where x denotes the owner of the key, symmetric keys as k_{xy}, and encryption and decryption of a message N (with slight but usual abuse of notation) as $c_x(N), d_x(N), k_{xy}(N)$, and $k_{xy}^{-1}(N)$, respectively. In the concrete parts, we assume that RSA [15] and a symmetric system with 128-b keys like IDEA [16] are used. We assume OFB mode, i.e., a pseudo one-time pad. Thus synchronization as supported by ISDN is a precondition for the protocol, such that each arriving bit can be decrypted at once. The participants must not get out of synch. Keeping this in mind, the usage of a synchronous stream cipher provides for very fast mixing and can be implemented without testing for replays.

We use *hybrid encryption of minimal length:* If Alice, A, wants to send a message N to Bob, B, she first generates a key k_{AB}. She appends as much of N (N^+) to k_{AB} as fits in the same block of the asymmetric encryption system and encrypts this block with c_B. Finally, she encrypts the rest of N (N^{++}) with k_{AB}. We denote the entire operation by

$$c_B^*(N) = c_B\big(k_{AB}, N^+\big), k_{AB}\big(N^{++}\big). \tag{3.1}$$

The randomly chosen key k_{AB} also makes RSA probabilistic, and to prevent attacks from [17], k_{AB} and the rest of the first block should be mingled by symmetric encryption with a globally known key.

We prescribe that A uses a fixed *mix cascade*, say $mix_1, \cdots, mix_m$, for all her channels. This does not reduce anonymity; quite the reverse. If no cascade is used, someone who communicates with her more than once might intersect these anonymity sets. Mix cascades also reduce the problem

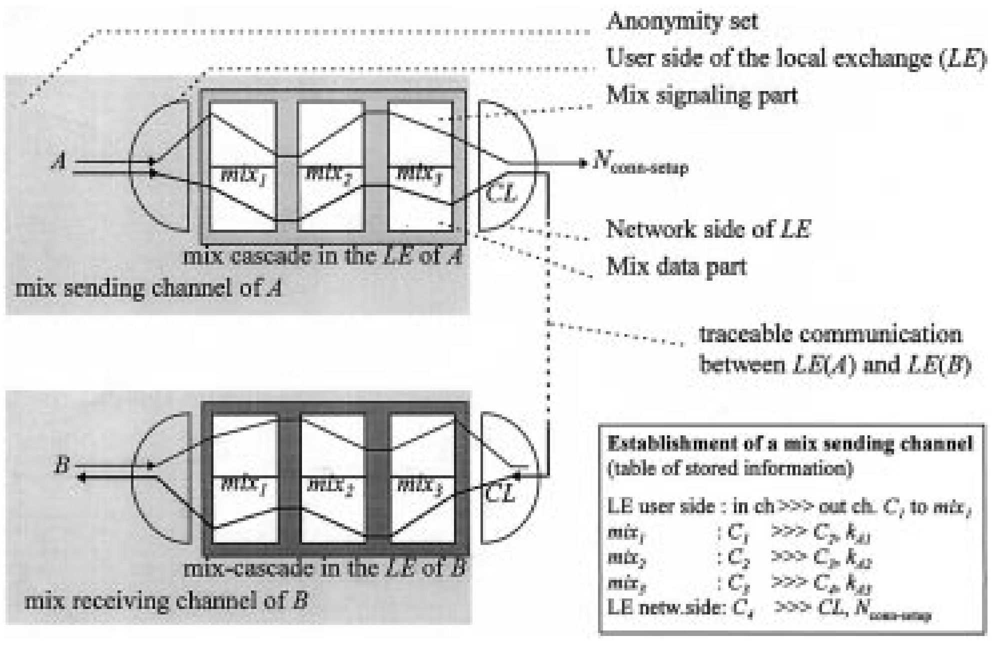
Anonymity set
User side of the local exchange (LE)
Mix signaling part
A
mix1
mix2
mix3
CL
N conn-setup
Network side of LE
Mix data part
mix cascade in the LE of A
mix sending channel of A
traceable communication between LE(A) and LE(B)
B
mix1
mix2
mix3
CL
mix-cascade in the LE of B
mix receiving channel of B
Establishment of a mix sending channel
(table of stored information)
LE user side : in ch >>> out ch. C1 to mix1
mix1 : C1 >>> C2, kA1
mix2 : C2 >>> C3, kA2
mix3 : C3 >>> C4, kA3
LE netw.side: C4 >>> CL, N conn-setup

Fig. 1. A mix channel.

that all messages of a batch must be of equal length and timing problems between the mixes; both are critical for performance. In the ISDN scenario, the cascade will be situated at A's LE, i.e., between user side and network side of the LE. Each mix_i for $i \geq 2$ initially generates a key pair (c_i, d_i) and publishes c_i, and A shares a symmetric key k_{A1} with mix_1.

With hybrid encryption, we can use the basic mix scheme for very long messages. However, this is not enough for our purpose. Each bit on the data channel should be handled at once, because each mix would still have to wait for the whole first block of data to arrive before it can start decrypting. Therefore we make the asymmetrically encrypted part a separate message that will be sent on the signaling channel before the actual data transmission starts. The actual data can then be transmitted in the data channel without any bandwidth expansion and without delay.

We only consider simplex channels; duplex channels will be realized as a pair of simplex channels. Moreover, we distinguish *mix sending channels* that keep the sender anonymous and *mix receiving channels* that keep the recipient anonymous; they will be combined later.

1) Mix Sending Channels: The sender A constructs the *setup message* for a mix sending channel as follows, where N are signaling data that should be output by the last mix of the cascade and D_i are data that A wants to give to mix_i in addition to the symmetric key k_{Ai}

$$\begin{aligned} N_{m+1} &:= N \\ N_i &:= c_i^*(D_i, N_{i+1}) \quad \text{for } i = m, m-1, \cdots, 2 \\ N_1 &:= k_{A1}(D_1, N_2). \end{aligned} \tag{3.2}$$

She sends N_1 to the user side of her LE, which passes it to mix_1. Each mix_i receives N_i from mix_{i-1} or A and decrypts it with d_i or k_{A1}, respectively. It processes the data D_i in whatever way is intended and forward N_{i+1} to mix_{i+1} or, for $i = m$, further in the network. In the ISDN scenario, this will mean back to the LE for routing in the long-distance network. In our case, we simply have $D_i := t_i$ where t_i is a timestamp. In contrast to the original mixes, this eliminates the need to compare messages of different batches for repeats. Because of the fixed mix cascades, the timestamps only have to be local sequence numbers of batches of setup messages.

Each mix_i for $i < m$ processes this setup message as follows; see Fig. 1 (what mix_m does will be described in the context of the final scheme).

1) It reserves an outgoing data channel C_{i+1} to mix_{i+1} for the following continuous data. It is sensible to sort the outgoing channels in the same way as the corresponding setup messages in the output batch.
2) It tells the position of the outgoing channel to mix_{i+1}, together with the decrypted setup message N_{i+1}.
3) It stores the correspondence between the incoming and the outgoing channel.
4) It stores k_{Ai} as belonging to this correspondence. (Of course, the index "A" is only notation; no mix after mix_1 knows that the key is from A.)

2) Mix Receiving Channels: To make recipients anonymous with respect to senders, we use mix-receiving channels. They have a setup message constructed just like that for a mix-sending channel (except for the part N which was left open anyway).

3) Connecting the Two Halves: To make A and B both anonymous, we will connect a mix-sending channel from A with a mix-receiving channel built up by B. The basic idea is the innermost parts, N, of both setup messages contain a common *channel label*, CL, and, on A's side, routing information to the mix cascade B uses. In the ISDN context, this will be the address of B's local exchange. How A and B agree on the label will be shown in the final scheme. We call the result a mix channel which is shown in Fig. 1.

4) Mix Data Channels: Recall, mixing setup messages, and mixing data will be different modules. After establishing a sending channel, mix_i can immediately decrypt each bit of data arriving on the incoming channel with k_{Ai} and forward it on the corresponding outgoing channel. In Fig. 1, this is represented as the lower half of the mixes.

For the receiving channel, the corresponding data channel is established and used in the reverse way: Channels from mix_{i+1} to mix_i are reserved. In particular, the last channel leads from mix_1 to B. When data arrive on the data channel, mix_i encrypts them (not "decrypts" as with a mix-sending channel) with k_{Bi}. Thus B receives multiple-encrypted data and decrypts them with all his keys k_{Bi}. This is a variant of the anonymous return addresses in [2]; however, here B establishes a return channel himself instead of passing a return address to A. This is more efficient in our scenario and will foil active attacks on B by A.

5) Problems with Pure Mix Channels: The main problem with pure mix channels is how to get enough channels of equal length to be mixed together. For channels, this means that the setup messages belong to the same batch and the following continuous data start and end at the same time; otherwise an attacker could distinguish such channels in spite of mixing. It might be difficult enough to obtain enough connections starting at the same time, if connections must be established within, say, 3 s, without routing them unnecessarily through the long-distance network. And it is highly unlikely that enough of them end at the same time by themselves. Therefore, some users would have to wait for others before being allowed to release their channels (i.e., their NT's would send encrypted nonsense when the continuous data end). However, each user only has two channels; thus it cannot be tolerated that they are blocked.

B. Real-Time Mixes

In this section, we present the complete technique of Real-Time Mixes. It solves all the remaining problems of mix channels: How does B know that he should set up a mix receiving channel for A's data, and how does A know the label? How do we get enough channels at each local time that start together, and how can we make them end together? The solution consists of: 1) time-slice channels; 2) dummy traffic where it costs nothing; and 3) broadcast of short connection-setup messages.

1) Time-Slice and Dummy Channels: Each connection is divided into a sequence of time-slice channels, that look completely unrelated to everybody except for the respective sender and recipient. With each new time slice, users can release connections and/or establish new ones. Each participant who does not need a channel during a time slice establishes a dummy time-slice channel instead. This costs no additional bandwidth because this channel is on the subscriber line only.

More precisely, during each time slice, each subscriber A maintains two mix-sending channels and two mix-receiving channels through the m mixes at her local exchange. They are called time-slice sending (TS-) channels and time-slice receiving (TR-) channels and the corresponding setup messages *TS-and TR-setup messages*. Thus, before each time slice, A must send two TS- and two TR-setup messages.

The last mix at A's local exchange passes the innermost part, N, of each setup message to the local exchange for connection. Fig. 2 shows simplified the setup messages for a connection establishment. For any TR-setup message, N only consists of a connection label CL. Any TS-setup message contains a connection label CL, the address A_{LE} of the local exchange of the corresponding recipient and information about the message to pass as well as the connection-setup message itself, i.e., $N_{\text{conn-setup}}$. The information indicates the status of the setup message, i.e., meaningful or meaningless (mf_m, ml_m), and the data, e.g., dummy traffic is meaningless (ml_D). If A has a real connection with a recipient B, the connection labels must agree. This will be achieved by letting A and B generate them pseudorandomly from the same seed CL_{init}, which will be exchanged in $N_{\text{conn-setup}}$ described below. Recall that a real connection is duplex and realized by two independent simplex channels.

If B has no real connection on one of its two duplex channels during a time slice, he sets up a simplex channel with himself instead. That is, he sends a TS-setup and a TR-setup message with the same connection label, e.g., at time slice t_1 $CL_B(j)$ (see Fig. 2). Hence these dummy channels look exactly like local connections. This makes local connections completely unobservable except for the initial wish to establish one. In this case the mf_m/ml_m-field indicates this as a real, meaningful message. For long-distance connections, one can only observe which anonymity sets communicate.

The data in the data channel are treated according to the setup messages as described under mix channels, i.e., forwarded from the sending channel to the specified receiving channel (real or dummy), with one exception: During connection establishment, when the sender already designates a real recipient in the TS-setup message but does not yet send real data (compare with t_1 and t_2), we will allow her to specify that data as meaningless (ml_D). Thus the data (that must be mixed anyway) are thrown away on the network side instead of being unnecessarily transported through the long-distance network. If at any mix, no data arrive on a channel, the mix replaces them by random data so that no one can simply see where a channel leads by cutting it off.

2) Establishing Connections: If A wants to call B, she needs to tell him that he should stop setting up dummy channels with himself and set them up to meet A's channels instead. Thus one *connection-setup message* $(N_{\text{conn-setup}})$ must be delivered to B in a different way. Still, B may not want A to know his real location address inside his anonymity set. We achieve this by allowing *implicit addresses* and broadcasting short connection-setup messages in B's anonymity set, i.e., to all subscribers at B's LE.

An implicit address of B allows B, but nobody else, to identify a message as belonging to him; see [1] for a systematic treatment. The typical cases are the following.

a) *Invisible Public Addresses:* Here, with B's public key, A encrypts the message and some redundancy that can be recognized by B's NT. This address, i.e., the encrypted message, is called invisible because others cannot even

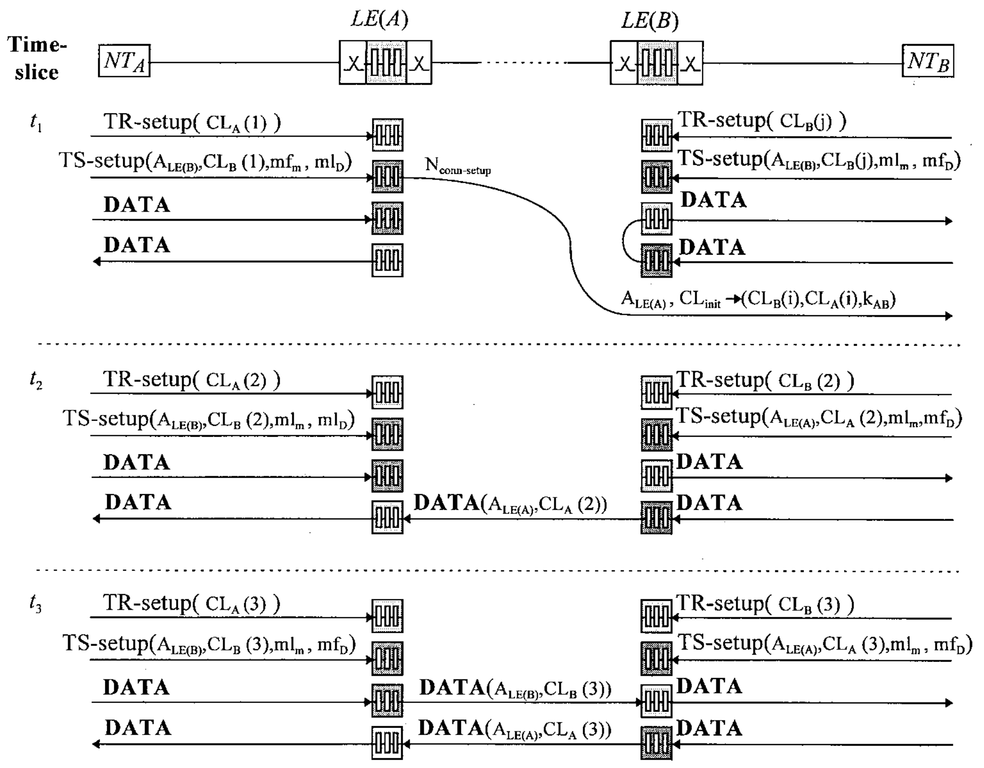

Fig. 2. Simplified connection establishment in the easiest case. See the surrounding text for notations and Section IV-C for the exact syntax description.

identify whether two messages have the same address. Hence the key to build such an address, i.e., the key c_B, can be put into directories. Everybody can link the key to the user or the pseudonym but not to the real identity. For example, for Alcoholics Anonymous: B wants to call Alcoholics Anonymous, their public key is available in the "Yellow Pages." B generates his message and sends it to his own $LE(B)$ which routes it to LE (Alcoholics Anonymous), i.e., the anonymity set where this organization belongs to. There the message is broadcast. Since the address of B's LE is included, the anonymous alcoholics organization may reply to $LE(B)$ on the prepared channel without revealing B's anonymity.

b) *Visible Private Addresses:* These are simply randomly chosen labels; thus the NT's of the potential recipients can evaluate them much faster. They should only be used once. For example, such an address could be used if B called A before and now wants to be called back.

We will only consider invisible addresses further. Apart from or rather within the implicit address, $N_{\text{conn-setup}}$ must contain:

a) the address of A's local exchange, so that B can address his TS-channels there;
b) a seed CL_{init} from which A and B will generate the connection labels for the future time slices of this connection with a pseudorandom generator;
c) the number t_i of the time slice so that B knows that this is when A starts running the pseudorandom generator.

Per time slice, one label is generated for the simplex channel from A to B and one for that from B to A. We call them $CL_B(i)$ and $CL_A(i)$, respectively, where i stands for the number of the time slice within the connection. $N_{\text{conn-setup}}$ may also contain a key k_{AB} for hybrid encryption as usual, or such a key can be derived from the seed before the labels.

On A's side, $N_{\text{conn-setup}}$ is packed into the innermost part N of a TS-setup message. Thus we have no problem of finding enough other connection-setup messages to mix it with. Now all TS-setup messages must contain such a field so that they are of equal length. Recall, the additional short field mf_m/ml_m indicates whether this field is a real, meaningful connection-setup message or not.

A successful connection establishment is shown in Fig. 2 and described in the following. For each user only one of the two duplex channels with signaling and data part is shown.

a) Time slice 1: Both A and B have no real connection. A wants to establish one to B. Thus she indicates this in her TS-setup message. Because of mf_m, $N_{\text{conn-setup}}$ is recognized by the network side of $LE(A)$ and therefore, it is sent in the network and broadcast as connection wish by $LE(B)$. All subscribers of $LE(B)$ test whether the received message is addressed to them. Only the network termination NT_B of B can recognize $N_{\text{conn-setup}}$ which A has encrypted with c_B. After decryption, B finds the CL_{init} for the connection labels (and a symmetric key). As B does not know anything about A's connection wish at the beginning of this time slice, he sets up dummy channels to himself as before, i.e., B's NT^B sends dummies on his time-slice channels to himself, using a random connection label $CL_B(j)$. The field "ml_m" in the TS-setup messages characterizes his connection-setup message field as meaningless. Analogous the field "mf_D/ml_D" characterizes whether meaningful or meaningless data are sent on the data channel. A's TR-setup message is random and the data channels is explicitly denoted as meaningless, i.e., ml_D, so that it is not routed through the long-distance network following the TS-setup message. To keep the transparency of the data channel the mf_D/ml_D-bit is implemented in the TS-setup message.

b) Time slice 2: The originator of the connection, A, cannot send any meaningful data yet because she does not know whether B accepts the connection. Hence, ml_D is used in t_2. However, her TR-setup message sets up her TR-channel so that she can receive data from B, i.e., with the specified connection label [in our example $CL_A(2)$]. As B has a free channel, B's NT accepts the connection and builds up a TS-channel to A, using the address $A_{LE}(A)$ from the received connection-setup message and generating CL_A and CL_B from CL_{init} for the current time-slice, i.e., $CL_A(2)$. At the same time, B's phone starts ringing, and the data transmitted from NT_B to NT_A tell A this fact. When B picks up the receiver, his NT signals this fact to A in the data channel. NT_A and NT_B are synchronized.

c) Time slice 3 (and following): Now both communication partners build up their time-slice channels with the corresponding connection labels and exchange data.

2) Releasing Connections: The LE's should not release channels in the long-distance network after every time slice. Instead, they can either wait until the channel is not needed again in the next time slice, or (with slight restriction of bit transparency) the end of the connection can be signaled in the data channel. Thus for each real connection, e.g., a phone call, a channel through the long-distance network is switched once at most.

If a NT does not answer at once, it is no problem for anonymity because the mixes will fill up the sender's unused TR-channel.

In principle, A can let her NT wait whether B will answer for an arbitrary period of time. For example, if the connection-setup message contains an additional field which is shown to B. He might end his previous connection in favor of the new one. It is then useful to include an upper bound on the time A will wait after sending the connection-setup message so that B will not be disturbed, nor will NT_B build up a long-distance channel to NT_A after A gave up waiting. Moreover, A in person should not be forced to wait on the phone, but be notified by a short ring when B answers. Then A will typically not have to repeat her connection-setup message to B. This significantly reduces the broadcast traffic on the signaling channel.

Remarks: If we allow such long waiting times, the connection labels should be generated by a pseudorandom function of the time-slice number instead of sequentially. In practice, such a function can be derived from the symmetric encryption system used.

Recall that the data on the data channels are sent continuously, i.e., the setup messages for time slice i are sent on the signaling channel at the same time as the data of time slice $i-1$ are sent on the data channel. In this respect, the illustration of the timing in Fig. 2 is a bit simplified.

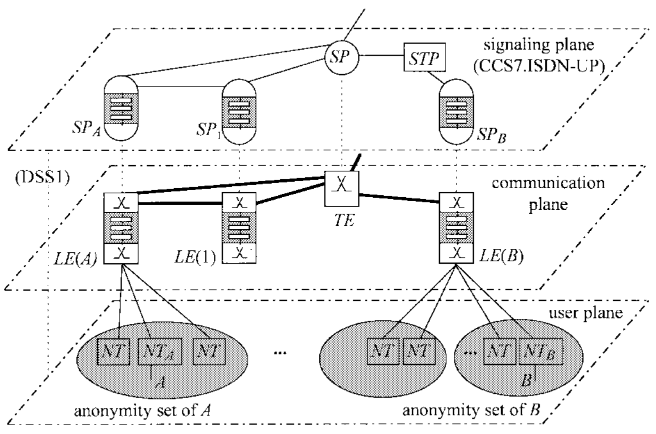

Fig. 3. Basic ISDN structure and modifications.

IV. Specification of the Anonymity Protocols for ISDN

We now show how the abstract Real-Time Mixes from Section III can be mapped onto an ISDN network.

A. Basic Signaling Structure of the ISDN

Signaling is the transfer of information for controlling purposes. In the ISDN, signaling and data channels are separated throughout. The signaling structure of the ISDN is commonly described as in Fig. 3. The signaling plane consists of signaling points (SP), corresponding to the LE's or transit exchanges (TE) of the communication plane, and signaling transfer points (STP). The user plane contains the network terminations (NT) and the participants.

Two different signaling protocols are used: The Common Channel Signaling System no. 7 (CCS7) in the signaling plane, and the Digital Subscriber Signaling System no. 1 (DSS1) at the user-network interface, i.e., for signaling between NT's and LE's on the 16-k bit/s signaling channel. CCS7 has several functional parts [18]. The Message Transfer Part, corresponding to layers 1 to 3 in the ISO OSI reference model, offers a universal transport system for signaling messages. Above it are different so-called user parts (UP) for different services. The one for the ISDN is called ISDN-UP. It additionally uses a Signaling Connection Control Part. However, we only need to modify the ISDN-UP. The ISDN protocol DSS1 covers layers one to three of the OSI model. We only have to modify layer three.

B. Overview of the Modifications to the ISDN

In the network structure, the only new components are mix cascades associated with local exchanges. Virtually, they are separated into a signaling and a communication part; this was already shown in Fig. 3. Both the first and the last mix are connected to the local exchange, so that all the usual network functions are still handled by the LE's. Physically, the mixes are of course not "in" the local exchange as the figures might suggest, because they need independent owners, but they need dedicated lines. Moreover, the performance analysis will show that we can only support anonymity sets of about 5000 users. Thus, if more NT's are connected to a local exchange, they

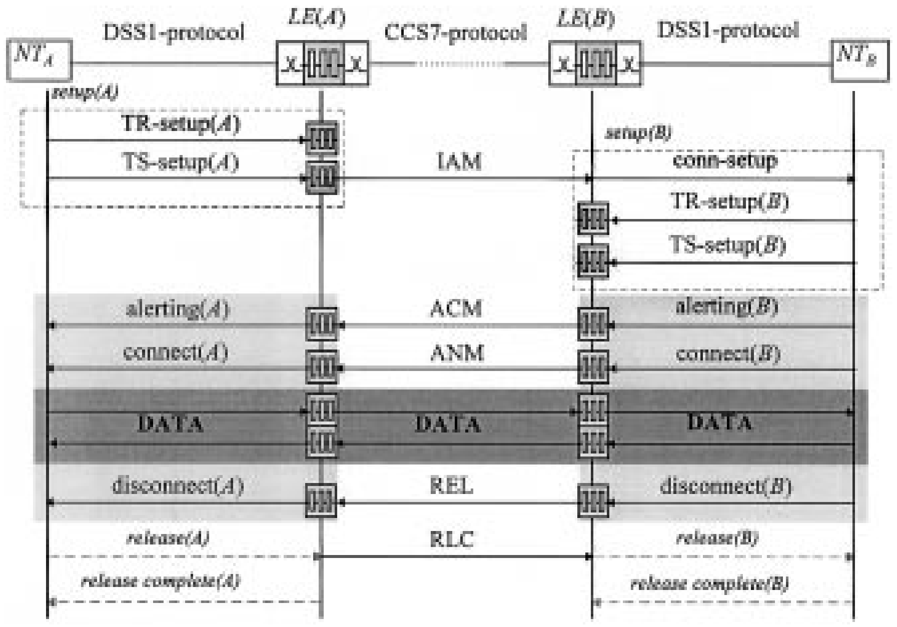

Fig. 4. Basic ISDN connection establishment protocol and modifications.

will be virtually partitioned into such sets, i.e., mixing and broadcasting connection-setup messages will only take place within such a set.

We will not work out detailed protocols for the signaling among mixes and between mixes and LE's where we are free, but concentrate on what needs to be modified in the existing ISDN protocols on the existing lines, where we have strict constraints. As sketched above, we only have to change CCS7.ISDN-UP and layer three of DSS1. In particular, routing, network management, and the execution of CCS7 signaling messages are unchanged. Fig. 4 gives an overview of connection establishment in the ISDN, covering both protocols, and our modifications.

The alerting and connect messages notify A that B's phone is ringing and that B picked up the receiver, respectively. They are sent on the data channel, represented by the light gray area in Fig. 4. This is reasonable because these messages must be anonymous and other ways of sending them would cost more of the critical bandwidth of the signaling channel; the bit transparency is not restricted because the connection is still unused. The corresponding CCS7 messages, however, are sent in the signaling plane as usual.

The disconnect message, that tells one partner that the other put down the receiver, is treated similarly. This is a small restriction of bit transparency. One alternative is to handle a few bits of the signaling channel as an "extension" of the data channel. The messages release and release complete are not needed in the local exchange area because the connection is released automatically with every time slice. Releasing a connection is only necessary in the long-distance network. The individual messages are described separately in the following sections.

C. The Parameters and Their Length

We need the length of the mix-specific parameters to see whether they fit into the frames of the given ISDN protocols (and for the performance analysis below). The length of the ciphertext of a message N in hybrid encryption of minimal length (see Section III-A) is

$$|c_B^*(N)| = \max(b_{\text{asym}}, |N| + s_{\text{sym}})$$

where b_{asym} is the block length of the asymmetric encryption system and s_{sym} the key length of the symmetric system. For $i < m$, all N_{i+1} contain at least one asymmetrically encrypted block, thus $|N| >= b_{\text{asym}}$. The length of a mix input message N_1 (3.2) is:

$$|N_1| = \max(b_{\text{asym}}, |D_m| + |N| + s_{\text{sym}}) + \sum_{j=1}^{m-1} |D_j| + (m-2)s_{\text{sym}} \tag{4.1}$$

for $m \geq 2$ mixes. In the following, we use parameter lengths as given in Table I. With this and if D_i is simply a time-slice number, we obtain

$$|N_1| = \max(660, |N|+158) + (m-1)30 + (m-2)128. \tag{4.2}$$

D. Modification of the DSS1 Protocol

The structure of a DSS1 signaling message in layer 3 is shown in Fig. 5.

1) Message Types and Information Elements: The message head of a DSS1 message has a length of 32 b. The codes of the message types and information elements are given in [19, Figs. 5.19 and 5.20]. For the anonymity protocol, we need to modify their codes (see Tables II and III).

Time-slice channels work without receipts, therefore we do not use message types like call proceeding, connect acknowledge, and setup acknowledge. As mentioned (see Fig. 4), the release and release complete messages can also be omitted. The information message is not needed, because all connection information is already included in the setup messages. Progress

TABLE I
LENGTH OF PARAMETERS (ISDN-SPECIFIC ONES ACCORDING TO [19])

name of parameters	abbreviation	length	name of parameters	abbreviation	length
time slice number	t	30 bit	channel label	CL	28 bit
initial value of the channel label	CL_{init}	128 bit	meaningful/meaningless identifier	mf/ml	1 bit
block of the asymmetric system	b_{asym}	660 bit	address of a local exchange	A_{LE}	14 bit
key of the symmetric system	s_{sym}	128 bit			

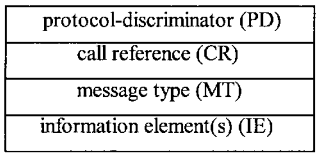

Fig. 5. General structure of a DSS1 signaling message.

TABLE II
MODIFIED CODES OF THE DSS1 MESSAGE TYPES OF THE BASIC SEQUENCE

message type	code	message type	code
TR-setup	0000 0101	alerting	0000 0001
TS-setup	0000 1101	connect	0000 0111
connection-setup	0000 0010	disconnect	0100 0101

and notify are not necessary either. Omitting all these messages reduces the data on the signaling channel, which is critical for performance.

The codes of existing message types and information elements (IE's) are maintained (see Table III). New elements obtain codes which were or became free. Some of the IE's originally belonged to the signaling channel. They are still required, but shifted to the data channel. The following IE's *must* be omitted for anonymity: calling party number, calling party subaddress, called party number, and called party subaddress. The following IE's are omitted because they are not needed or not used by the PTT's: notification indicator, display, date/time, and sending complete. The new IE's TS- and TR-information, as well as the new MT connection-setup contain the cryptographic data of our TS- and TR-setup and connection-setup messages, respectively. The IE's alerting information and connected information are mix information that need to be sent with alerting and connect/disconnect messages. The coding rules are described in [19] and [20].

2) Complete Signaling Messages of the DSS1: We now show the entire structure of the new message types.

a) TR-setup: This message is used to set up the time-slice receiving channel between a NT and a LE. It contains the TR-setup message in the sense of Section III as an IE. The innermost part N is a channel label CL. For example, for three mixes, $N_{\text{TR-setup}} = k_{A1}(t_i, c_{A2}(t_i, k'_{A2}, c_{A3}(t_i, k'_{A3}, CL_A(i))))$. By (4.2), the length of this information element in bits is

$$|N_1| = 660 + (m-1)30 + (m-2)128.$$

A complete TR-setup message is given in Table VI. The length of a DSS1 layer-three message is limited to 260 byte. Longer messages can be segmented according to [20, Annex K], but currently no European PTT supports this. Hence, either a segmentation procedure must be implemented or m, the number of mixes, must be small enough. In the latter for example, with eight mixes the TR-setup message is 224 byte long and the longest message, TS-setup, 250 byte.

b) TS-setup: This message is used to set up a TS-channel between a NT and a LE. Its IE contains the TS-setup message in the sense of Section III. According to Fig. 2 the innermost

TABLE III
MODIFIED CODES OF THE IE'S OF SEVERAL MESSAGE TYPES

information elements	code	information elements	code
alerting information	B-channel	bearer capability	0000 0100
cause	B-channel	channel identification	0001 1000
connected information	B-channel	recipient information	0110 1100
high layer compatibility	0111 1101	low layer compatibility	0111 1100
progress indicator	B-channel	TR-information	0110 1101
TS-information	0111 0000	shift	1001 xxxx

part N of $N_{\text{TS-setup}}$ received by the network side of $LE(A)$ is

$$N_B := A_{LE(B)}, CL_B(i), mf_m/ml_m, mf_D/ml_D, N_{\text{conn-setup}}.$$

By (4.2), the length of the resulting IE in bits is

$$|N_1| = 862 + (m-1)30 + (m-2)128.$$

c) Connection-setup: This message is broadcast to all the members of an anonymity set without passing through the mix cascade:

$$N_{\text{conn-setup}} := c_B(\text{connection-setup\_msg}, t_i, A_{LE(A)}, CL_{init}).$$

Hence

$$|N_{\text{conn-setup}}| = 660 \text{ b}.$$

d) Alerting message: These messages are sent in the data channel when the phone starts ringing at the recipient's side (see Fig. 4), so that the sender can get the ringing tone. The messages are

$$N_{\text{alerting}(B)} := k_{B1}(k_{B2}(\cdots k_{Bm}(\text{alerting\_msg}, A_{LE(A)})))$$
$$N_{\text{alerting}(A)} := k_{A1}(k_{A2}(\cdots k_{Am}(\text{alerting\_msg})))$$

where the keys are those from the corresponding setup messages. To handle such messages, LEs must read information on the data channel, and the NT's must not encrypt these signaling messages with the symmetric key k_{AB} from the connection-setup message. The message lengths are 22 and 8 b, independent of the number of mixes.

TABLE IV
CODES OF THE INFORMATION ELEMENTS

information elements	code	information elements	code
access transport	0000 0011	automatic congestion level	0010 0111
backward call indicator	0001 0001	call history information	0010 1101
cause indicators	0001 0010	connection request	0000 1101
echo control information	0011 0111	forward call indicators	0000 0111
generic notification	0010 1100	channel label	0000 0100
nature of connection indicators	0000 0110	optional backward call indicators	0010 1001
originated ISC point code	0010 1011	parameter compatibility info	0011 1001
propagation delay counter	0011 0001	system clock	0000 1010
transmission medium requirement	0000 0010	transm. medium requirement prime	0011 1110
transmission medium used	0011 0101	user service information	0001 1101
user service information prime	0011 0000	user teleservice info	0011 0100
user to user indicator	0010 1010	User to User Information	0010 0000

Connect and disconnect messages: They are sent if the recipient has picked up the receiver:

$$N_{\text{connect}(B)} := k_{B1}(k_{B2}(\cdots, k_{Bm}(\text{connect\_msg}, A_{LE(A)})))$$
$$N_{\text{connect}(A)} := k_{A1}(k_{A2}(\cdots, k_{Am}(\text{connect\_msg}))).$$

The disconnect messages signal the end of the communication. If B is the party who puts down the telephone receiver then

$$N_{\text{disconnect}(B)} := k_{B1}(k_{B2}(\cdots, k_{Bm}(\text{disconnect\_msg}, A_{LE(A)})0))$$
$$N_{\text{disconnect}(A)} := k_{A1}(k'_{A2}(\cdots, k'_{Am}(\text{disconnect\_msg}))).$$

Their natural lengths are 22 and 8 b; if we send them in the data channel they should be so that, given that the data channel is encrypted, the probability that they occur by chance is smaller than other error probabilities of the ISDN.

E. Modification of the CCS7 Protocol

1) Message Types and Information Elements: The functional structure of CCS7 and especially the tasks of the ISDN-UP were sketched in Section IV-A.

The message types (IAM, etc.) and their codes remain unchanged. The codes of existing IE's are also maintained according to [21, Table 5]. New elements obtain codes which were or became free (see Table IV).

In contrast to the original code, the data channel must now be switched symmetrically for realization of anonymous signaling.

2) Complete Signaling Messages in the CCS7 Protocol:

IAM: The IAM message is originated by the DSS1 TS-setup message. It signals the recipient B that somebody is calling him. Only B can notice this message after he has correctly deciphered the connection-setup message broadcast by $LE(B)$.

$$N_{\text{IAM}} := N_B = c_B(\text{connection-setup\_msg}, t_i, A_{LE(A)}, CL_{init}), A_{LE(B)}.$$

The modified structure of an IAM message as well as of the following messages is given in the Appendix. All maintained, discarded, and new parameters of the mandatory fixed, the mandatory variable and the optional parts of these messages (see Fig. 6) as well as their lengths are shown in Table VII. Some of the original parameters were discarded to guarantee anonymity of the subscribers.

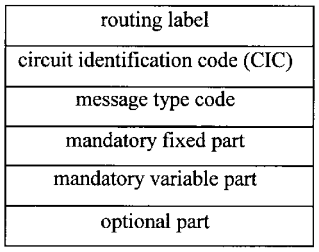

Fig. 6. General format of an ISDN-UP message.

It is no problem to send longer messages since message segmentation is implemented in CCS7.

ACM (address complete message): If an alerting message arrives, $LE(B)$ builds:

$$N_{\text{ACM}} := \text{acm\_msg}, t_i, A_{LE(A)}, CL_A(i), A_{LE(B)}.$$

Hereby, $LE(B)$ uses the stored data, i.e., the channel identifier of the receiving channel of the sender (cf. Table VII for the modified structure of ACM).

ANM (answer message): If a DSS1 connect message arrives, the $LE(B)$ signals (cf. Table VII) with

$$N_{\text{ANM}} := \text{anm\_msg}, t_i, A_{LE(A)}, CL_A(i), A_{LE(B)}$$

that the called party has picked up the receiver and the communication can start.

REL (release): Triggered by the DSS1 disconnect message, with

$$N_{\text{REL}} := \text{rel\_msg}, t_i, A_{LE(A)}, CL_A(i), A_{LE(B)}$$

it is signaled that the recipient has dropped the line and therefore ended the communication. Additionally, $LE(A)$ knows that the data channel in the long-distance network can be released (cf. Table VII).

RLC (release complete): This message has not been modified by the proposed protocol. It is used as a receipt of the REL message where $LE(B)$ knows that the data channel is now released. RLC contains only the parameter cause identifier that describes the purpose of release.

TABLE V
LENGTH OF THE MODIFIED ISDN MESSAGES IN COMPARISON

	Original length	length of the new IEs with 10 mixes	modified length with 5 mixes	modified length with 10 mixes
setup	1084 bit	–	–	–
TR-setup	–	1954 bit	1532 bit	2322 bit
TS-setup	–	2156 bit	1734 bit	2524 bit
conn-setup	–	660 bit	1028 bit	1028 bit
alerting	296 bit	22 bit	320 bit	320 bit
connect	404 bit	22 bit	394 bit	394 bit
disconnect	204 bit	22 bit	230 bit	230 bit
IAM	976 bit	674 bit	1166 bit	1166 bit
ACM	1076 bit	94 bit	458 bit	458 bit
ANM	1216 bit	94 bit	466 bit	466 bit
REL	912 bit	94 bit	362 bit	362 bit

V. PERFORMANCE

The main question of the performance analysis is whether the signaling channels between the NT's and LE's can support all our new time-slice setup messages and the broadcast of connection-setup messages. All the other changes have no significant effect upon performance.

A. Comparing the Length of the Modified ISDN Messages

Table V shows the length of all modified and their corresponding original messages. The original lengths of DSS1 and ISDN-UP signaling messages were determined from [20] and [21] (first row). Section IV D and E discussed the parameters of mix input and signaling messages. From this the length of the new IE's (second row) is calculated for ten mixes. The lengths of the complete modified signaling messages are then given in row three and four (for five and ten mixes), that is, the results of the fourth row are determined by adding the second row and the results of the first row after all maintained, discarded, and new parameters from the original message were considered.

B. Minimal Duration of a Time Slice

A time slice must be long enough so that the four time-slice setup messages can be sent on each local signaling channel.

The total bandwidth of the signaling channel, in each direction, is 16 kb/s. Some of it must be used for error control, several narrowband services and other signaling messages. We restrict us and use only 12 kb/s for connection-establishment messages. Since the signaling channel is a full duplex channel, broadcasting the connection-setup messages does not conflict with the setup messages. As stated in Section III, a recipient has to test every connection-setup message. This does not mean testing the whole message, rather he attempts to decrypt the first asymmetric block of every connection-setup message. The resulting overhead is tolerable since asymmetric encryption handles hundreds of k bit/s but there are only 12 kb of data per second.

Each NT must be able to transmit two TR- and two TS-setup messages at its signaling channel, hence

$$z \geq \frac{2 \cdot (|N_{\text{TR-setup}}| + |N_{\text{TS-setup}}|)}{12 \text{ kbit/s}}.$$

Therefore

$$z \geq \frac{2 \cdot (1954\text{b} + 2156\text{b})}{12000\text{b/s}} = 0.685\text{s}$$

if ten mixes are used for every cascade. Thus, a lower bound on the length z of a time slice is 0.7 s. If five mixes are used, then $z \geq 0.421$ s. See Fig. 7 for more cases. If a time-slice is 0.7 s, a subscriber must wait on average 0.35 s before he can set up a connection.

C. Size of an Anonymity Set

The size of an anonymity set is limited by the distribution of the connection-setup messages. To evaluate this, data about busy hour, connection-establishment attempts are needed. We use a very conservative bound of a worst-case average of 12 connection-setup messages to each subscriber in every hour, i.e., an arrival rate of $\lambda = 1/300\ \text{s}^{-1}$. Every connection-setup message is 660 b long, independent of the number of mixes. At most $\mu = b/N_{\text{conn-setup}}$ messages can be broadcast per second where b is the available bandwidth. For $b = 12$ kb/s, this is $\mu = 18.18\ \text{s}^{-1}$.

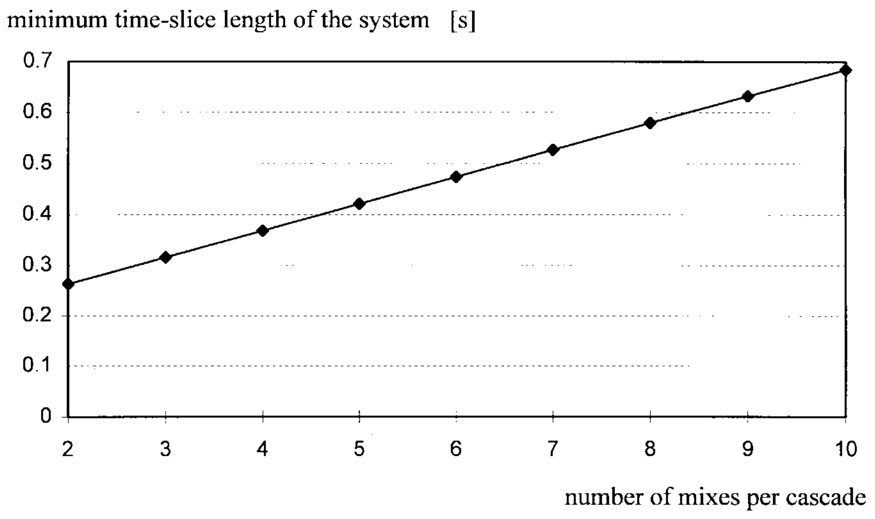

Fig. 7. Dependence of the minimal time slice length and the number of mixes used.

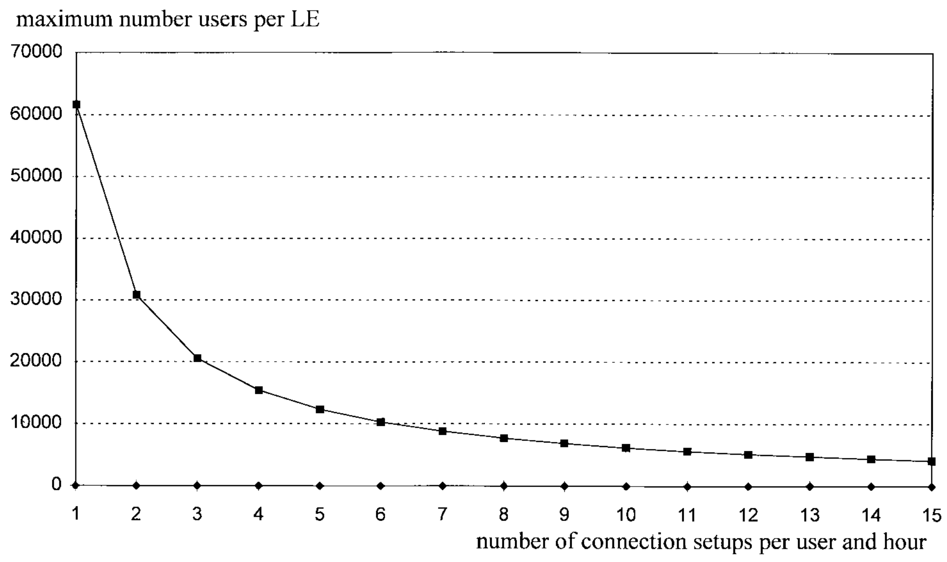

Fig. 8. Relation of user traffic and the maximum size of the anonymity set.

The system is approximately an M/D/1 queueing system. The system time, i.e., the average time a connection-setup message needs after arriving at $LE(B)$ is then (see [22])

$$T_v = \frac{2 \cdot \mu - n \cdot \lambda}{2 \cdot \mu \cdot (\mu - n \cdot \lambda)}.$$

If we require an upper bound of $T_v \le 0.5$ s, we obtain $n \le 5137$ for the size of an anonymity set. More values are shown in Fig. 8. If more users are connected to a LE the upper bound of T_v increases. Hence, the average number of connection wishes must be reduced.

The number of anonymity sets is restricted by the number of LE's which can be supported. For addressing a LE, 14 b are allowed by the ISDN standard.

D. Connection-Establishment Time

The time needed for the computations, both at the terminal equipment and at the mixes can be made almost negligible, say 0.01 s, if we use sufficiently parallel hardware in the mixes. Thus the time until a connection from A to B has been established mainly consists of the following parts, if B accepts this connection at once:

1) waiting until A's mix cascade mixes TS-setup messages ($\le z$);
2) delay time of the mix cascade of $LE(A)$ ($m \cdot 0.01$ s);
3) connection establishment in the long-distance network (≤ 0.2 s);
4) waiting until the connection-setup message is broadcast to B (in peak hours $T_v \le 0.5$ s on average);
5) waiting until B can establish a mix channel with A ($\le z$);
6) delay time of the mix cascade of $LE(B)$ ($m \cdot 0.01$ s).

The sum is $2(z + m \cdot 0.01 \text{ s}) + T_v + 0.2$ s. With the figures used above, in particular $z = 0.7$ s, this is about 2.3 s. On average, the connection establishment will take about half as long; see Fig. 9.

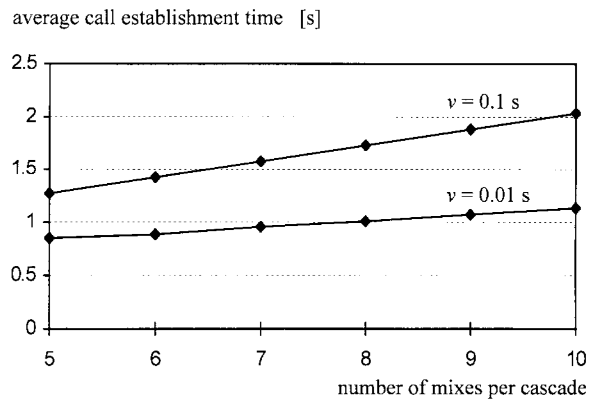

Fig. 9. Relation of the average connection-establishment time and the number of mixes per cascade for two computation times (v) per mix.

VI. Remarks on Active Attacks and Billing

A. Active Attacks

We have already prevented one active attack in Section III-B by letting the mixes fill up channels where no data arrives. We can also prevent an active attack on availability that is special to our protocol: An attacker could particularly easily clog the signaling channel by sending large numbers of senseless connection-setup messages. To prevent this, priority tokens for them could be introduced by which the LE of the recipient determines the broadcast order. No token is needed for the lowest priority. Each participant only obtains a limited number of high-priority tokens from the PTT. The tokens are signed by the PTT using blind signatures like in an anonymous payment systems [9], but free of charge.

Some more difficult active attacks, however, cannot be prevented completely. They all exploit situations where an attacker knows that an anonymous participant X should react in some way. The attacker then prevents a real participant A from reacting, e.g., by cutting her line. Thus, if X reacts, $X \neq A$ has been proved, and if X does not react, it is likely that $X = A$.

If mix_1 or an attacker on A's subscriber line modifies the data on this line and colludes with B, they can test whether A is B's current communication partner, because B will receive nonsense in this case. A similar attack is to block A's resources, e.g., by establishing two nonanonymous connections with A. If the attacker can then establish a connection with X, too, then $X \neq A$. Such attacks cannot be prevented. However, they need the collusion of the communication partner of X and one must disturb lots of participants to carry them out. Hence this seems acceptable in practice.

Another attack is to disrupt the broadcast of connection-setup messages: Assume $LE(A)$ or an attacker on the subscriber lines causes an connection-setup message to be handed to A only. Then, if the connection is accepted, A must be the recipient. This attack can be detected if each network termination sends a digital signature of (a hash value of) all messages received on the broadcast channel back to each mix, e.g., once per time slice. If a mix detects an inconsistency, one can try to localize the attacker; the mixes must also sign their outputs for this purpose.

TABLE VI
Complete TR-Setup Message

Bit No.:	8	7	6	5 4 3 2 1
Byte No.: 1	protocol discriminator (0000 1000)			
Byte No.: 2..3	call reference (0000 0001) (0000 1000)			
Byte No.: 4	message type (0000 0101)			
Byte No.: 5..8	0	BC I.E. identifier (000 0100)		
	length of BC contents (0000 0010)			
	1	0	0	ITC(speech) (0 0000)
	0/1	1	0	ITR (1 0000)
Byte No.: 9..10	0	CHI I.E. identifier (001 1000)		
	B-channel identifier + excl/pref			
Byte No.: 11..13	low layer compatibility I.E.			
Byte No.: 14..16	high layer compatibility I.E.			
Byte No.: 17..19	progress indicator I.E.			
Byte No.: 20..264	time slice receiving information I.E.			

So far we deliberately ignored the problem that a user has to trust her or his NT to perform the anonymizing operations correctly and not to reveal secrets to unauthorized parties. In a real-world implementation, this implies, e.g., that these parts of the NT's must not be controlled by the network operator, but by the users themselves.

B. Billing

For local connections, i.e., inside an anonymity set, flat-rate accounting could be used because they do not use shared resources. Then billing is necessary for long-distance connections only. Billing anonymous connections can be done via local counters in the network terminations or an anonymous payment system [9], [23]. In the latter case, the subscribers buy so-called stamps from the PTT from time to time, which are special messages signed by the PTT using blind signatures. These stamps can be included in the TS-setup messages, for example. For equal length, all TS-setup messages then need a stamp field, but a meaningful stamp is only needed if a long-distance channel is set up, i.e., if the address A_{LE} is non-local and the mf_D bit for the data is set.

VII. Conclusion

We have presented techniques extending mixes for efficient continuous, real-time communication.

They can be used for various communication systems. As a practical example, we have shown that anonymous telephony is feasible under the technical assumptions made by the PTT's for the narrowband ISDN. The service considered in this paper

TABLE VII
MODIFIED STRUCTURE OF THE CCS7 MESSAGES ACCORDING TO [19], AND [21]

msg	new parameters	maintained parameters	discarded parameters
IAM	user to user inf. (674 bit)	nature of connection ind. (8 bit)	user to user inf. (24-1048 bit)
		forward call indicators (16 bit)	calling party category (8bit)
		transmission medium req. (8 bit)	closed user group lock code (48 bit)
		connection request (56-72 bit)	calling party number (32-96 bit)
		access transport (>24 bit)	redirecting number (32-96 bit)
		user service inf. (32-104 bit)	redirection inf. (24-32 bit)
		user to user indicator (24 bit)	called party number (32-88 bit)
		propagation delay counter (32 bit)	MLPP precedence (8 bit)
		user serv. inf. prime (32-104 bit)	original called number (32-96 bit)
		originated ISC point code (32 bit)	generic number (40-104 bit)
		user teleservice info (24 bit)	location number (40-96 bit)
		parameter comp. info (>32 bit)	
		generic notification (24 bit)	
		transm. medium req. pr. (24 bit)	
ACM	system clock (30 bit)	backward call indicator (16 bit)	user to user inf. (24-1048 bit)
	channel label (28 bit)	optional backw. call ind. (24 bit)	access delivery inf. (24 bit)
	user to user inf. (12 bit)	cause indicators (>32 bit)	redirection number (40-96 bit)
		user to user indicator (24 bit)	call diversion inf. (>32 bit)
		access transport (>24 bit)	redirection number restr. (24 bit)
		generic notification (24 bit)	
		transm. medium used (24 bit)	
		echo control information (24 bit)	
		parameter comp. inf. (>32 bit)	
ANM	system clock (30 bit)	backward call indicator (32 bit)	connected number (32-96 bit)
	channel label (28 bit)	optional backw. call ind. (24 bit)	generic number (32-96 bit)

(a)

is the creation of a 64-k b/s duplex channel between network terminations with connection establishment times of about 3 s. Participants are anonymous within fixed anonymity sets of about 5000 subscribers, i.e., connections can only be traced to such a group. The performance analysis shows that the costs of the changes to the ISDN network structure and its signaling procedures are acceptable.

We did not describe additional ISDN features such as call forwarding, but most of them could also be realized without giving up anonymity. Recall, however, that we described accounting procedures, and that anonymous communication does not complicate identification on higher layers, e.g., by digital signatures, where that is needed.

Remaining problems are that it may not be trivial to organize the responsibility for the mixes so that every participant trusts at least one mix including its hardware and operating system, and that there are no efficient countermeasures against some specific active attacks on anonymity. Moreover, the security

TABLE VII (*Continued.*)
Modified Structure of the CCS7 Messages According to [19], and [21]

		user to user indicator (24 bit)	user to user inf. (32-1048 bit)
		access transport (>24 bit)	access delivery inf. (24 bit)
		generic notification (24 bit)	redirection number (40-96 bit)
		parameter comp. inf. (>32 bit)	call diversion inf. (24 bit)
		call history information (32 bit)	redirection number restr. (24 bit)
		transm. medium used (24 bit)	
		echo control information (24 bit)	
REL	system clock (30 bit)	cause indicators (>24 bit)	user to user inf. (32-1048 bit)
	channel label (28 bit)	access transport (>24 bit)	redirection number restr. (24 bit)
		automatic congest. level (32 bit)	user to user indicator (24 bit)
		parameter comp. inf. (>32 bit)	access delivery inf. (24 bit)

(b)

of all mix schemes relies on computational assumptions. However, active attacks directed against specific participants are also possible outside a network, and computational assumptions are also used with other security primitives. Hence the security achieved seems quite satisfactory for practical purposes.

Let us conclude that, although communication technologies and services develop very fast, it is possible to maintain user privacy, and that the task of doing so should not be neglected.

Appendix

See Tables VI and VII.

Acknowledgment

The authors would like to thank M. Böttger, H. Federrath, Prof. Dr. W. Görke, M. Seeger, and the anonymous referees for helpful discussions.

References

[1] A. Pfitzmann and M. Waidner, "Networks without user observability—Design options," *Eurocrypt '85*, 1986, pp. 245–253 (Extended version in *Computers & Security 6/2*, 1987, pp. 158–166).

[2] D. Chaum, "Untraceable electronic mail, return addresses, and digital pseudonyms," *Commun. ACM*, vol. 24, no. 2, pp. 84–88, 1981.

[3] D. Cooper and K. Birman, "The design and implementation of a private message service for mobile computers," *Wireless Networks*, vol. 1, pp. 297–309, 1995.

[4] L. Cottrell, "Frequently asked questions about mixmaster remailers, FAQ version 1.8 (July 1996)." [on line]. Available WWW: http://www.obscura.com/^\simloki/remailer/mixmasterfaq.html.

[5] C. Gülcü and G. Tsudik, "Mixing e-mail with Babel," presented at the Symp. Network and Distributed System Security 1996, San Diego, CA, Feb. 22–23.

[6] H. Federrath, A. Jerichow, and A. Pfitzmann, "Mixes in mobile communication systems: Location management with privacy," in *Proc. Inform. Hiding*, 1996, pp. 121–135.

[7] J. Müller, "Anonyme Signalisierung in Kommunikationsnetzen," Diplomarbeit, TU Dresden, Instit. für Theoretische Informatik, 1997.

[8] P. Syverson, D. Goldschlag, and M. Reed, "Anonymous connections and onion routing," in *Proc. 1997 IEEE Symp. on Security and Privacy*, Oakland, CA, pp. 44–54.

[9] D. Chaum, "Security without identification: Transaction systems to make big brother obsolete," *Commun. ACM*, vol. 28, no. 10, pp. 1030–1044, 1985.

[10] ——, "The dining cryptographers problem: Unconditional sender and recipient untraceability," *J. Cryptology*, 1/1, pp. 65–75, 1988.

[11] A. Pfitzmann, "A switched/broadcast ISDN to decrease user observability," in *Proc. 1984 IEEE Int. Zurich Seminar on Digital Communications*, Zurich, Switzerland, pp. 183–190.

[12] ——, "Diensteintegrierende Kommunikationsnetze mit teilnehmerüberprüfbarem Datenschutz," IFB 234. Heidelberg, Germany: Springer-Verlag, 1990.

[13] M. Reiter and A. Rubin, "Crowds: Anonymity for web transactions," DIMACS, Tech. Rep. 97-15, Apr. 1997 (revised June 1997).

[14] B. Schneier, *Applied Cryptography*. New York: Wiley, 1995.

[15] R. Rivest, A. Shamir, and L. Adleman, "A method for obtaining digital signatures and public-key cryptosystems," *Commun. ACM*, vol. 21, no. 2, pp. 120–126, 1978.

[16] X. Lai and J. L. Massey, "A proposal for a new block encryption standard," in *Eurocrypt '90*, pp. 389–404, 1991.

[17] B. Pfitzmann and A. Pfitzmann, "How to break the direct RSA-implementation of MIXes," in *Eurocrypt '89*, pp. 373–381, 1990.

[18] A. R. Modarressi and R. A. Skoog, "Signalling system no. 7: A tutorial," *IEEE Commun. Mag.*, vol. 28, no. 7, pp. 19–35, 1990.

[19] G. Bandow, H. Gottschalk, D. Gehrmann, W. Hlavac, H. Koch, W. Müller, and D. Schwetje, "Zeichengabesysteme—Eine neue Generation für ISDN und intelligente Netze," *L.T.U. Vertriebsgesellschaft*, Bremen, Germany, 1995.

[20] ETSI, ISDN, DSS1 protocol, "Signalling network layer for circuit-mode basic call control, Part 1 Protocol specification," Nov. 1995.

[21] ITU-T: Q.763 Specification of Signalling System No. 7, Formats and Codes of the ISDN User Part of Signalling System No. 7; Mar. 1993.

[22] L. Kleinrock, *Queueing Systems—Volume 1: Theory*. New York: Wiley, 1975.

[23] D. Chaum, "Privacy protected payments—Unconditional payer and/or payee untraceability; SMART CARD 2000: The future of IC cards," in *Proc. IFIP WG 11.6 Int. Conf.*, Laxenburg, Austria, 19.-20. 10. 1987, pp. 69–93.

[24] A. Pfitzmann, B. Pfitzmann, and M. Waidner, "ISDN-MIXes-untraceable communication with very small bandwidth overhead," in *Proc. Kommunikation in Verteilten Systemen, IFB 267*, 1991, pp. 451–463.

[25] ——, "Telefon-MIXe: Schutz der Vermittlungsdaten für zwei 64-kbit/s-Duplexkanäle über den $(2*62+16)$-kbit/s-Teilnehmeranschluß," *Datenschutz und Datensicherung DuD*, pp. 605–622, 1989.

Anja Jerichow received the diploma in computer science from the University of Technology, Dresden (TUD), Germany, and the M.S. degree in computation from Oxford University, U.K..

Since 1994, she has worked as a Research Assistant at TUD. Her main research interests include security and privacy, particularly in communication networks and mobile computing. Currently, she works mainly on theoretical aspects of mixes.

Jan Müller is a student of computer science at the University of Technology, Dresden, Germany.

His research interests include privacy and security issues, mainly in communication networks and mobile computing.

Andreas Pfitzmann (S'83–A'83–M'90) received the diploma and Ph.D. degrees in computer science from the University of Karlsruhe, Germany.

He is a Professor of computer science at the University of Technology, Dresden, Germany. His research interests include privacy and security issues, mainly in communication networks, mobile computing, and distributed applications such as payment systems and has authored or coauthored about 50 papers in these fields.

Dr. Pfitzmann is a member of ACM, and GI, where he serves as chairman of the Special Interest Group on Dependable IT-Systems.

Birgit Pfitzmann received the diploma from the University of Karlsruhe, Germany, and the Ph.D. degree from the University of Hildesheim, Germany, in computer science in 1990 and 1994, respectively.

She is currently a Professor of computer science at the University of Saarbrücken, Germany. Her main research interests are cryptography, security, and privacy, particularly in electronic marketplaces, and specification of distributed systems. She is the coauthor of about 50 publications and takes part in practical projects sponsored by the European Union.

Michael Waidner received the diploma and Ph.D. degree in computer science from the University of Karlsruhe, Germany, in 1986 and 1991, respectively.

He is the Manager of the network security research group at the IBM Zurich Research Laboratory, Switzerland. His research interests include cryptography, security, and all aspects of dependability in distributed systems. He has coauthored numerous publications in these fields.

Dr. Waidner is a member of ACM, the GI, the IACR, and SIAM.