to appear in the Proc. of EUROCRYPT '89, LNCS, Springer-Verlag, Berlin 1990

HOW TO BREAK THE DIRECT RSA-IMPLEMENTATION OF MIXES

Birgit Pfitzmann, Andreas Pfitzmann

Institut für Rechnerentwurf und Fehlertoleranz, Universität Karlsruhe
Postfach 6980, D-7500 Karlsruhe 1, F. R. Germany

ABSTRACT

MIXes are a means of untraceable communication based on a public key cryptosystem, as published by David Chaum in 1981 (CACM 24/2, 84-88) (=[6]).

In the case where RSA is used as this cryptosystem directly, i.e. without composition with other functions (e.g. destroying the multiplicative structure), we show how the resulting MIXes can be broken by an active attack which is perfectly feasible in a typical MIX-environment.

The attack does not affect the idea of MIXes as a whole: if the security requirements of [6] are concretized suitably and if a cryptosystem fulfils them, one can implement secure MIXes directly. However, it shows that present security notions for public key cryptosystems, which do not allow active attacks, do not suffice for a cryptosystem which is used to implement MIXes directly.

We also warn of the same attack and others on further possible implementations of MIXes, and we mention several implementations which are not broken by any attack we know.

I. INTRODUCTION: MIXES

Basically, a MIX-network [6] is a means of sender anonymity, which can at the most be computationally secure.

Meanwhile, other sender anonymity schemes have been published, which are information-theoretically secure, namely superposed sending (DC-net) [7, 8] and, against more limited attackers, RING-networks [18]. Nevertheless, MIXes are still a matter of interest, since their communication overhead is much smaller. More precisely, in the other schemes, each participant has to send about as much in the physical sense as all the participants together want to send in the logical sense. With MIXes, the overhead is at most about the product of what the participant himself wants to send and the number of MIXes he uses; for long messages in some MIX-schemes there is nearly no overhead. Therefore, MIXes seem to be the only way to provide sender anonymity for telephony using the cables of conventional telephone networks, i.e., the only way complete privacy can be introduced in public communication networks in the near future [20].

The idea behind MIX-networks is that a, hopefully trustworthy, station called MIX collects a number of messages from their senders, performs a cryptographic operation on each of them to change their outlooks, and outputs them to their addressees in a different order. Thus an attacker should not be able to find out which outgoing message corresponds to which incoming one (except possibly his own).

Of course, the recipients should be able to read their messages in spite of the change in outlook. Therefore, in the basic scheme (which is part of nearly all more sophisticated schemes), the MIX achieves this change by deciphering using a public key cryptosystem, and the senders must prepare their messages by enciphering them with the public key of the MIX. (If every message passed only one MIX, the scheme could be changed so that the MIX would use a private key with each sender. But usually, several MIXes in a row are used for every message, because in this case it suffices that one of them is trustworthy, and those in the middle may know neither sender nor recipient.)

So far, since the encryption function was assumed to be deterministic in [6], like that of RSA, everybody could take an output message, encrypt it again and check which input message they obtain. To avoid this reencryption attack, nondeterminism was introduced by attaching a random part to each message before encryption. After decryption, the MIX deletes this part. Another simple attack, replay, was avoided by having the MIX discard repeated input messages. Otherwise, an attacker could find out what becomes of an input message by repeating it and observing which output message is repeated.

To avoid confusion about which kinds of MIXes we break and which not, we distinguish (top-down) the basic idea of a MIX-network, MIX-schemes (e.g. the described basic scheme, or the return address scheme [6]), implementations of MIX-schemes (e.g. additional use of redundancy), and the choice of the cryptosystem to be used. (The "unsealing" operation in [6] is just the cryptographic operation the MIX performs in the basic MIX-scheme, "sealing" the corresponding operation the sender performs.)

In [6], the random part is always implemented as a large random string in front of the message. (This does not imply that no other operations can be performed, but that they would have to be considered as a part of the cryptosystem; this is no loss in generality if the requirement "$K(K^{-1}(X))=X$", which implies bijectivity, is dropped. For our purposes it is better to discuss such operations seperately as implementation.)

As an example of a public key cryptosystem, RSA is mentioned, and at that time there seems to have been little choice.

Without knowing about further attacks, one might have considered it natural to use RSA directly as the cryptosystem in the implementation with the random string in front. We call this the *direct RSA-implementation of MIXes*. For this case, the operation of a MIX with the modulus $m = p \bullet q$, the public exponent c for enciphering, and the private exponent d for deciphering is shown in Figure 1. N_1 and N_2 are two messages, R_1 and R_2 the attached random strings, and the commas denote concatenation.

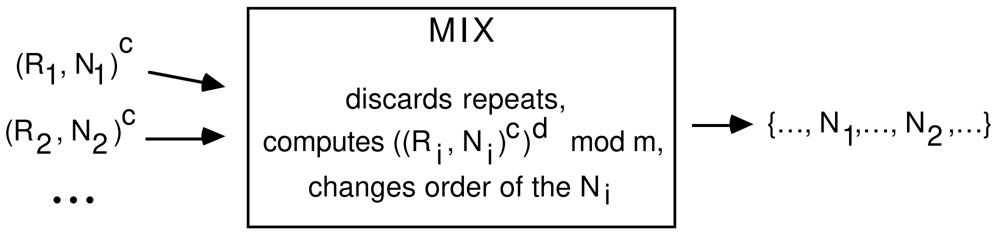


Figure 1 Operation of a MIX of the basic scheme in its direct RSA- implementation

II. THE ATTACK

The attack as described in this section is specific for the direct RSA-implementation of the basic MIX-scheme (cf. Section I). In Section III we discuss which other possible schemes, implementations, and cryptosystems are vulnerable.

II.1. HISTORY

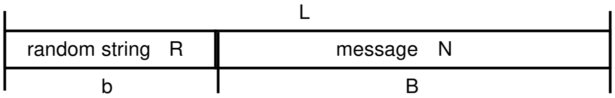
The attack is based upon the well-known attack on RSA, which exploits the fact that RSA is a multiplicative homomorphism, by Davida [9] in the version by Judy Moore (according to [11]).

It has been adapted to other situations before (no guarantee on completeness): to cryptosystems with some abstract properties in [10, 17], to signatures with some redundancy in [16], or to yield a factoring algorithm [12]. It was put to positive use for blind signatures and, thereby, for untraceable credentials and payments (e.g. [7]). It also forms a small substep in proofs of the security of single bits of RSA, from [15] to [1]).

II.2. THE IDEA

In our case, the difficulty with the well-known RSA-attack lies in the fact that the random string in a decrypted message is not output, and that an attacker who forms his message according to the attack, instead of according to the MIX-scheme, does not know which output corresponds to his own input. (This last fact is also the reason why the system is not trivially broken by the active attacks of [15], for which an oracle outputting the last bit of a message suffices.)

All following congruences are modulo m. $L := \lceil \log_2(m) \rceil$ is the block length of the cipher. Let the first b bits be reserved for the random strings and the remaining B bits for the messages, i.e. a message looks like this:



Thus the encrypted form M of a message N with attached random string R is

$$M \equiv (R \cdot 2^B + N)^c \bmod m.$$

Consider an attacker who wants to trace such a message M which was input to the MIX (see Figure 2). He chooses a "small" (cf. II.4.) factor f, forms $M^* :\equiv f^c \cdot M$ and inputs M^* to the MIX. On the one hand, the MIX decrypts M^* and interprets it as a message N^* with an attached random string R^*, i.e. as

$$M^{*d} \equiv R^* \cdot 2^B + N^*,$$

of which it outputs N^*.

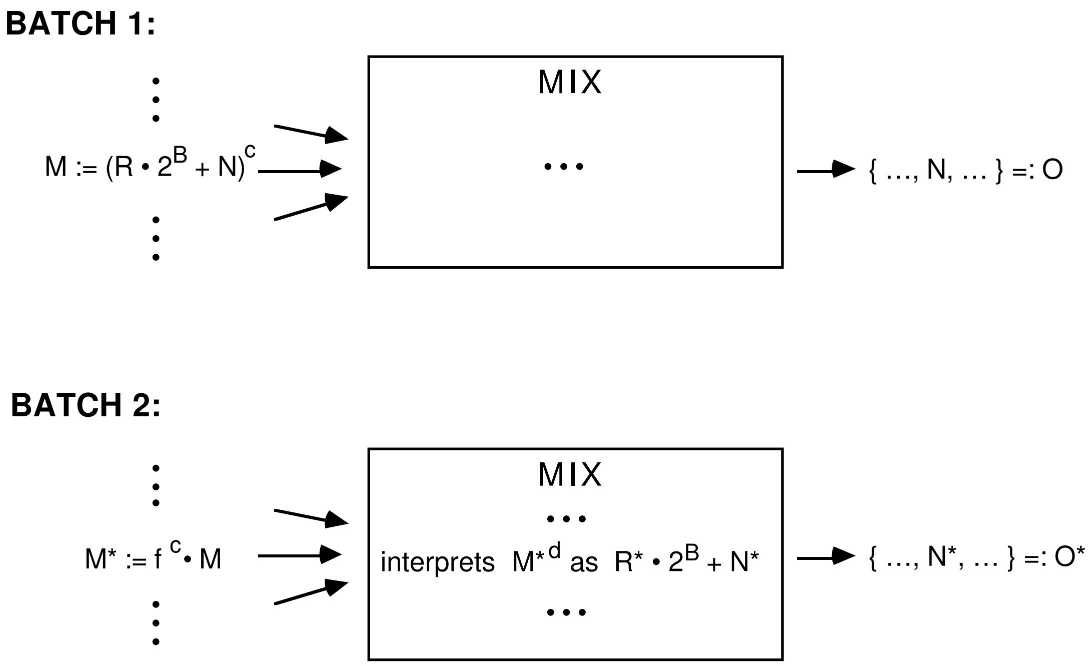


Figure 2 Scenario of the attack; the two batches are not necessarily distinct

On the other hand, the attacker knows that

$$M^{*d} \equiv f^{c \cdot d} \cdot M^d \equiv f \cdot (R \cdot 2^B + N).$$

Together this implies $N^* - f \cdot N \equiv f \cdot R \cdot 2^B - R^* \cdot 2^B$. Since $\gcd(2^B, m) = 1$ and 2^B therefore has an inverse modulo m, this becomes

$$(N^* - f \cdot N) \cdot 2^{-B} \equiv f \cdot R - R^*. \qquad (❋)$$

On the one hand, the attacker knows f and 2^{-B}, and that N and N* are in the sets O and O* of output messages of the respective batches. Hence if the batch size is reasonably small (this is always true in practice, cf. II.3.), he can compute all possible left sides of Equation (❋), i.e. all $(N_2 - f \cdot N_1) \cdot 2^{-B}$ for $N_1 \in O$, $N_2 \in O^*$.

On the other hand, $R, R^* \in \{0, \ldots, 2^b - 1\}$, thus for the right side of (❋)

$$f \cdot R - R^* \in \{-2^b+1, \ldots, f \cdot (2^b-1)\}.$$

The attacker now tries to find out which pair (N_1, N_2) of output messages is (N, N^*) by computing the left side of Equation (❋) for each of them and testing for this condition. If f has been chosen to be suitably small (cf. II.4.), the condition will not hold for most other values $(N_2 - f \cdot N_1) \cdot 2^{-B}$.

II.3. OVERHEAD

One can ask whether the batch size s can be made so large that the attacker cannot consider all pairs of output messages. For some time-critical services this seems impossible anyway, because there would not be enough messages to collect within the permitted time [19]. Anyway, the attacker can also speed up his attack by computing the sets of values which occur in the left sides of Equation (❋), i.e.

$$V_1 := \{f \cdot N_1 \cdot 2^{-B} \mid N_1 \in O\} \quad \text{(including } f \cdot N \cdot 2^{-B}\text{)}$$

and

$$V_2 := \{N_2 \cdot 2^{-B} \mid N_2 \in O^*\} \quad \text{(including } N^* \cdot 2^{-B}\text{)}$$

(all numbers reduced modulo m) and sorting them. Now he has to check for which $v_1 \in V_1$ there is a $v_2 \in V_2$ such that

$$v_2 - v_1 \in \{-2^b+1, \ldots, f \cdot (2^b-1)\} \bmod m. \qquad (✧)$$

This might look like Figure 3.

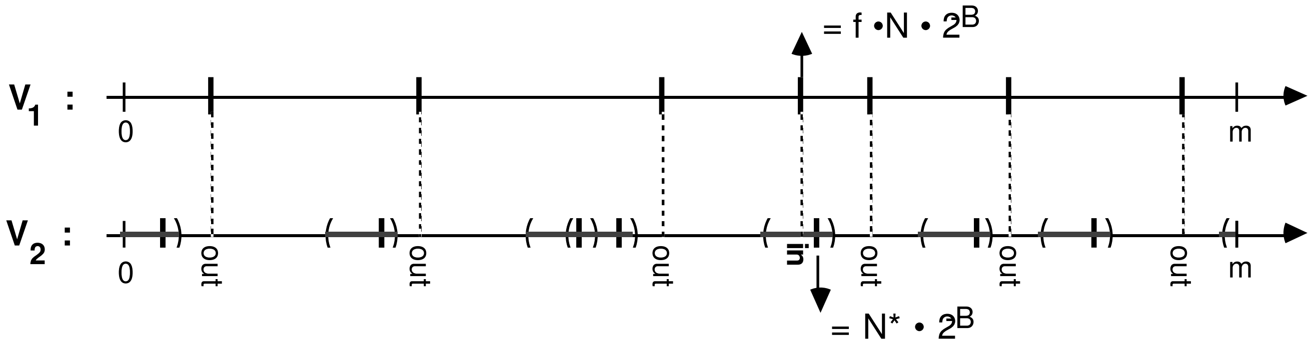


Figure 3 Test for Condition (✧); in the depicted case the attack is completely successful

All this can be done in O(s • log(s)) time.

So the attacker mainly has to carry out 1 modular exponentiation and division, 3•s modular multiplications, and to sort s numbers modulo m twice. The MIX itself has to carry out s modular exponentiations and to sort s numbers modulo m once (and many more, if the key is not changed with every new batch). Thus the complexity of the attack is nearly the same as that of legal mixing. Therefore the attack is always feasible.

II.4. CHOOSING THE FACTOR f AND THE PROBABILITY OF SUCCESS

The attacker is completely successful in identifying the message N, if $f \cdot N \cdot 2^{-B}$ is the only v_1 to fulfil Condition (✧). (It does not matter whether $N^* \cdot 2^{-B}$ is also the only v_2, because the attacker need not identify N^*.)

Assume that the values $v_1 \in V_1$ and $v_2 \in V_2$, except $N^* \cdot 2^{-B}$, were chosen from $\{0,\ldots, m-1\}$ randomly and independently. The probability that a specific $v_1 \neq f \cdot N \cdot 2^{-B}$ is near any of the v_2's in the sense of (✧), i.e. in the union of the intervals around them (cf. Figure 3), is then bounded by

$$\mathrm{Prob}(v_1) < s \cdot |\{-2^b+1,\ldots, f\cdot(2^b-1)\}| \;/\; m = s \cdot ((f+1)\cdot 2^b - f) \;/\; m$$
$$\approx s\cdot f\cdot 2^b \;/\; 2^{b+B} = s\cdot f / 2^B.$$

The messages N_1, N_2 can be assumed to be end-to-end encrypted and thus to be chosen randomly and independently. However, they are only chosen from $\{0,\ldots, 2^B-1\}$. Therefore the assumption is not really justified. Nevertheless, not knowing better, we will maintain it and hope that it is not too unreasonable, since the messages are multiplied by the rather large number 2^{-B} (it is at least $\approx 2^b$). The probability of complete success, i.e. that none of the s-1 other v_1's fulfils the condition, is then bounded by

$$\mathrm{Prob}(\text{success}) > 1 - (s-1)\cdot\mathrm{Prob}(v_1) \approx 1 - s^2\cdot f / 2^B.$$

Note that this probability is independent of the length b of the random string. Hence if B is large, say $B > 200$, the attacker can choose any factor $f < 2^{100,}$ and success will be quite certain since the batch size s can be assumed to be smaller than 10^{10}.

Thus those implementations that seemed most efficient when only the previously known attacks were considered, are insecure: One would have chosen $L > 600$ to ensure the security of the RSA modulus against factoring, and would have considered $b \approx 100$ sufficient to prevent an attacker from trying all random strings in a reencryption attack. The remaining $B > 500$ bits would have been used for the messages.

The attack can still succeed if, in most cases, more than one v_1 fulfils Condition (✧): The attacker can choose several factors f to attack the same message. For each f, messages $N_1 \in O$ which cannot be N according to (✧) can be excluded from further consideration. In this paragraph, we denote everything which corresponds to a given factor f by an additional subscript f.

Assume that the attacker inputs each M^*_f in a different batch. Then the corresponding sets $V_{2,f}$ are independent except for the values $N^*_f \cdot 2^{-B}$. Thus the conditions (✧) for the different $v_{1,f}$'s arising from a fixed $N_1 \neq N$ are nearly independent. The probability that the attacker cannot exclude N_1 from the possible values of N is therefore the product of the probabilities $\mathrm{Prob}(v_{1,f})$ over all chosen values f. He can minimize it by choosing as many small f's as possible, i.e. $f := 2, 3,\ldots$. If 2^B is not much larger than s, he must stop before $f > 2^B/s$.

The attacker can do even better. If (✧) holds for a given $v_{1,f}$, this is usually the case for just one $v_{2,f}$. He can use this $v_{2,f}$ to compute a tighter bound on R, which in turn makes the interval in (✧) smaller for the following factors f: If $v_{1,f}$ and $v_{2,f}$ really correspond to N and N^*, resp., then $v_{2,f} - v_{1,f} \equiv f \cdot R - R^*_f$. This gives $f \cdot R \equiv v_{2,f} - v_{1,f} + R^*_f$. Let x denote the right side of this congruence reduced mod m, i.e. $0 \le x < m$. Since $0 \le f\cdot R < m$, too (otherwise (✧) would always be true), the equation $f \cdot R = x$ holds in $\mathbb{Z}$. This means $R = x / f$. Since only R^*_f is unknown in x, x lies in an interval of size 2^b. Thus the attacker obtains an interval of size $2^b/f$ in which R lies. This knowledge can be used in (❋) to improve (✧).

To be safe from this attack, $\mathrm{Prob}(v_1)$ must be about 1 even for small f. This means approximately $2^B < s$. Choosing B so small in the basic MIX-scheme would not only reduce efficiency, the main advantage of MIXes, but is rather senseless, because it means that in every batch all possible messages occur, or the same message occurs several times. Thus the recipients would receive no information (in other words:

mostly, not even addressing would be possible, to say nothing of any message content). Other MIX-schemes where this does not hold are mentioned in Section III.

II.5. FEASIBILITY OF THE ACTIVE PART OF THE ATTACK

In principle, MIXes are better candidates for active attacks than most other users of a public key system for two reasons: First, they are forced to output much larger parts of messages they decrypted. Secondly, these messages are meaningless to them, because they are usually end-to-end-encrypted. Hence a MIX cannot detect an active attack by means of natural redundancy in the messages, as other users sometimes might. Nevertheless, there could be two problems for the attacker.

First, the attacker must see the message M he wants to trace before he can form a suitable M*. In analogy to the terminology of [14], one could call this a *directed* chosen ciphertext attack, directed against a particular input message.

If the MIX uses the same key for many batches, this is no problem. If a new key is used for each new batch, M* must be submitted to the MIX in the same batch as M. If the input messages are really collected, i.e. if they can arrive at any time and are stored by the MIX, the attacker still has a lot of time for his attack, at least against the early messages. (This would not hold for an *adaptive* chosen ciphertext attack, where the attacker would need an output from the MIX to form another input, e.g. if one could transform one of the attacks in [15] into a MIX-attack).

The attack can be prevented (without changing the scheme, implementation or cryptosystem of the MIX), if the MIX-network is changed so that the participants themselves store their messages and send them to the MIX at a prearranged time. But care must still be taken because the attacker only needs to perform one modular multiplication between seeing M and submitting M*. Hence if the messages arrive via the same time-division network, or from different local networks using different local clocks, the time might still suffice. So all the participants must be forced to send (or to commit to, cf. [5]) considerable parts of their messages before the first one has completed his. Also, this measure can no longer be applied if messages pass several MIXes: Here, each MIX must be considered as a potential attacker against its successor (this is why several MIXes are used), and of course it receives the messages early enough to perform an attack on them.

Secondly, if the MIXes are arranged as cascades in the sense that several MIXes must be passed in fixed order, only the first MIX is accessible to normal users directly. But this does not prevent an active attack on one of the others. Firstly, the previous MIX is a potential attacker (see above). Secondly, an attacker can prepare a message to attack the i-th MIX so that it passes through the previous i-1 MIXes correctly.

III. VARIATIONS

III.1. OTHER MIX-SCHEMES

The same attack can be used against the basic return address scheme [6, p.85] (if the same implementation and cryptosystem are used), because there, the address part of a message is just like a message in the basic scheme.

Many other schemes are based upon the idea in [6, p. 87] to use the basic scheme only for the first block of a message input to a MIX, and to include a private key in this block. The MIX uses this key to change the outlook of the rest of the message. Such schemes can enable better untraceability, increased performance, and fault tolerance [6, 19].

If, as in the original version in [6, p. 87], the address of the recipient or the following MIX is included in the first block, this address has the same effect as the message in the basic scheme. Thus the attack can be carried out. It will succeed if the batch size is usually smaller than the address space (cf. II.4.). This is quite likely e.g. if every station can act as a MIX. In this case, the attacker can even find out the enclosed secret key: Once he is able to trace messages, he can use the MIX as an oracle outputting the last bits of decrypted messages in Algorithm 1 of [15]. Of course, this only provides additional valuable information if the same key is to be used for future messages which have no obvious connection with the one already traced.

If the address is included in one of the other blocks, or if MIX-cascades must be passed in fixed order and therefore all but the last one output no address, this kind of attack is impossible.

Even if the implementation of the basic scheme used for the first block is secure, the danger of another, quite trivial, attack on such a scheme should not be overlooked: The secret key should not belong to a block cipher in electronic code book mode. Otherwise, patterns of equal blocks within a message would be repeated in the output. This is not necessarily checked by the MIX, because only repeats of the first blocks need to be discarded to prevent the replay attack (cf. Section I), and even if the senders took care to avoid this situation, an active attacker could replace some blocks in a message by repeats of others.

III.2. OTHER IMPLEMENTATIONS OF THE BASIC SCHEME

The same kind of attack is applicable (even easier), if the random string is attached at the end of the message instead of in front: A message is prepared as $M \equiv (N \bullet 2^b + R)^c$, and instead of Equation (❋), one has

$(N^* - f \bullet N) \bullet 2^b \equiv f \bullet R - R^*$.

A weaker form of the attack is still possible if the random part appears as bits inserted at predefined positions between the message bits, and either the random part is shorter than the message or there exists a large uninterrupted block of message bits:

Let "ins(R, N)" denote the operation of inserting the bits of R into N. An attacker who chooses M* as in II.2. knows that, similar to (❋), $\text{ins}(R^*, N^*) \equiv f \bullet \text{ins}(R, N) \bmod m$. He can transform this into equations in $\mathbb{Z}$, namely

$$f \bullet \text{ins}(R, N) \;=\; \text{ins}(R^*, N^*) + k \bullet m \quad \text{for some } k \in \{-1, \ldots, f-1\}.$$

If f is chosen to be a power of 2, $f \bullet \text{ins}(R, N)$ is just the bit pattern of ins(R, N), shifted to the left. If, moreover, f is very small, then for each $N_1 \in O$, $N_2 \in O^*$, and each possible k, the attacker can test whether it is possible that this equation holds: For fixed k he has a sum in $\mathbb{Z}$, where some bits are missing in two of the three numbers. He must test if the remaining patterns match.

Assume, e.g., that there is a block of 10 adjacent bits of N in ins(R, N), and that f=2. Then there are 9 adjacent bit positions where the bits of all the three numbers are known. Since there are two possible carries from the right, the probability that the equation holds on these bits is only 2^{-8} for randomly chosen N_1, N_2. In this way, each block of at least three adjacent bits can be tested.

Even if these tests don't rule out any message pairs and k's at once, the attacker can be successful. If, e.g., exactly every other bit is a message bit and f=2, then with each pair (N_1, N_2), the equation yields unique solutions R and R*. Using these, the attacker can reencrypt $\text{ins}(R, N_1)$ and check if it is M.

Nevertheless there are easy countermeasures to prevent this kind of attack. One possibility is to merge the random string and the message part completely before encryption, e.g. by encryption in another, unrelated cryptosystem (where the key need not be secret).

The other is to make the active attack infeasible by adding redundancy to the messages, as if active attacks on RSA itself are to be prevented (e.g. the image of N under a one-way function, again merged with the message somehow), so that the MIX will usually not accept $f^c \bullet M$ as properly formed.

Especially with MIXes, one could try to discard not only repeats, but also multiples of previous messages by small factors. "Is $M^* = f^c \bullet M$ for some small f?" can be tested as "Is $f := M^{*d} \bullet M^{-d}$ small?", thus with an expense of mainly one multiplication per message pair. If n is the number of messages mixed using the same key, this increases the complexity of mixing from $O(n \bullet \log(n))$ to $O(n^2)$ (ignoring log(m)-factors). For some services this is still feasible, at least if the key is changed with every new batch. For the basic scheme, "f is small" should be defined as "the check (✧) (cf. II.3.) provides information". Thus f is considered large enough if (✧) holds for all possible pairs (v_1, v_2). This means $(f+1) \cdot 2^b - f \geq m$, i.e. about $f > 2^B$. The probability that no legal messages collide in this sense is at least about $1 - n^2 \cdot 2^B/m \approx 1 - n^2/2^b$, if the $M^{d'}$s are chosen randomly (cf. II.4.). This is tolerable for some parameters. Nevertheless the more common redundancy schemes seem more convenient for both users and MIXes.

III.3. OTHER CRYPTOSYSTEMS

Also the direct use of another cryptosystem in the otherwise unchanged implementation would not necessarily be helpful (or in an implementation without the random string, if the cryptosystem is already probabilistic). Some public key cryptosystems which are provably secure against passive attacks are definitely not suitable, because they are known to be vulnerable to active attacks, and versions of these attacks can still be applied directly in the MIX-environment:

With the quadratic residuosity system of [13], the attack of Example 6 in [15], where the attacker inserts an encrypted bit of someone else's message into his own message in a disguised form, is possible.

With the system of [3] (the version as secure as factorization) one can apply one of the attacks which the authors themselves probably mean when stating that the system is insecure against active attacks: (Remember that encryption means that a pseudorandom number generator is run, which repeatedly squares a number and outputs the last bits each time. This output is added to the message from right to left. At the end, the following square is appended to the encrypted message, so that the recipient can recover the seed.)

To obtain the last bit of a certain square root of a given number z, the attacker uses z like the seed for the pseudorandom number generator, but squares z fewer times than prescribed. Now he adds the resulting (too short) bit string to the left part of an arbitrary message N (with or without an attached random string R, resp.). The MIX subtracts the same string again. Thus the attacker can recognize N by the unchanged left part, if he has chosen a sufficiently long string. The changes in the following log(log(m)) bits of N are the last bits of the square root of z. The last one of these was desired. Hence the attacker has a suitable oracle to factor using Algorithm 5 in [15]. (For an attack against the cryptosystem alone, all the squaring to recognize N is not needed.)

The second attack, being adaptive, can be avoided if the key is changed with every new batch. Redundancy in the messages alone would not help, because in each step the attacker can determine the message the MIX sees nearly completely. For the remaining log(log(m)) bits he can try every combination. This, in its turn, can be avoided by interdicting repeats of seeds for the pseudorandom number generator and of the intermediate squares.

There does not seem to be much use in discussing these special improvements to the implementation of the basic MIX-scheme further, because many seem secure against this special kind of attack and, at the moment, none of them seems provably secure.

Of course, the final consequence of discussions about both random strings and redundancy is that a public key cryptosystem used in the basic MIX-scheme should be probabilistic and secure against active attacks (cf. [2], but we have heard doubts about it; an interactive system, on the other hand, is quite unwieldy if several MIXes in a row are used [4]).

We are happy to thank Manfred Böttger and Michael Waidner for helpful comments, and Michael for lots of other support, too.

REFERENCES

[1] W. Alexi, B. Chor, O. Goldreich, C. P. Schnorr: RSA and Rabin functions: Certain parts are as hard as the whole; SIAM J. Comput. 17/2 (1988) 194-209.

[2] M. Blum, P. Feldman, S. Micali: Non-interactive zero-knowledge and its applications; 20th STOC, ACM, New York 1988, 103-112.

[3] M. Blum, S. Goldwasser: An Efficient Probabilistic Public-Key Encryption Scheme Which Hides All Partial Information; Crypto '84, LNCS 196, Springer-Verlag, Heidelberg 1985, 289-299.

[4] M. Böttger: Untersuchung der Sicherheit von asymmetrischen Kryptosystemen und MIX-Implementierungen gegen aktive Angriffe; Studienarbeit am Institut für Rechnerentwurf und Fehlertoleranz, Universität Karlsruhe 1989.

[5] G. Brassard: Modern Cryptology - A Tutorial; LNCS 325, Springer-Verlag, Berlin 1988.

[6] D. L. Chaum: Untraceable Electronic Mail, Return Addresses, and Digital Pseudonyms; CACM 24/2 (1981) 84-88.

[7] D. Chaum: Security without Identification: Transaction Systems to make Big Brother Obsolete; CACM 28/10 (1985) 1030-1044.

[8] D. Chaum: The Dining Cryptographers Problem: Unconditional Sender and Recipient Untraceability; J. of Cryptology 1/1 (1988) 65-75.

[9] G. Davida: Chosen Signature Cryptanalysis of the RSA (MIT) Public Key Cryptosystem; TR-CS-82-2, University of Wisconsin, Milwaukee (October 1982) (quoted in [11]).

[10] R. A. DeMillo, M. Merritt: Chosen Signature Cryptanalysis of Public Key Cryptosystems; Technical Memorandum, School of Information and Computer Science, Georgia Institute of Technology, Atlanta 1982. (quoted in [17]).

[11] D. E. Denning: Digital Signatures with RSA and Other Public-Key Cryptosystems; CACM 27/4 (1984) 388-392.

[12] Y. Desmedt, A. M. Odlyzko: A chosen text attack on the RSA cryptosystem and some discrete logarithm schemes; Crypto '85, LNCS 218, Springer-Verlag, Heidelberg 1986, 516-522.

[13] S. Goldwasser, S. Micali: Probabilistic Encryption; J. of Computer and System Sciences 28 (1984) 270-299.

[14] S. Goldwasser, S. Micali, R. L. Rivest: A Digital Signature Scheme Secure Against Adaptive Chosen-Message Attacks; SIAM J. Comput. 17/2 (1988) 281-308.

[15] S. Goldwasser, S. Micali, P. Tong: Why and How to establish a Private Code On a Public Network; 23rd FOCS, IEEE Computer Society, 1982, 134-144.

[16] W. de Jonge, D. Chaum: Attacks on Some RSA Signatures; Crypto '85, LNCS 218, Springer-Verlag, Berlin 1986, 18-27.

[17] M. John Merritt: Cryptographic Protocols; Ph. D. Dissertation, School of Information and Computer Science, Georgia Institute of Technology, February 1983.

[18] A. Pfitzmann: A switched/broadcast ISDN to decrease user observability; 1984 International Zurich Seminar on Digital Communications, IEEE, 1984, 183-190.

[19] A. Pfitzmann: How to implement ISDNs without user observability - Some remarks; Fakultät für Informatik, Universität Karlsruhe, Interner Bericht 14/85.

[20] A. Pfitzmann, B. Pfitzmann, M. Waidner: Datenschutz garantierende offene Kommunikationsnetze; Informatik-Spektrum 11/3 (1988) 118-142.