A Simple Technique for Diffusing Cryptoperiods

Stig F. Mjølsnes
Div. of Computer Science and Telematics
ELAB-RUNIT
The Norwegian Institute of Technology
N-7034 Trondheim, Norway

Abstract

The technique obtains diffuse cryptoperiods based on the stochastic properties of the cipher stream. The periods are randomized by scanning the pseudo-random bit sequence for occurrences of bit patterns. No explicit information about the change of key is necessary during transmission. The statistical model shows a deviation from the geometrical distribution due to overlapping between bit patterns. The technique can be generalized to randomize and synchronize any common event between sender and recipients without introducing extra signalling and with minimal computational overhead under the assumption of a reliable communication channel.

1 Introduction

In several key-management schemes proposed for conventional cryptosystems, the concept of a session key is prevailing [NS78, MM82, DP84]. Similar concepts are labeled data-encrypting key, primary communication key or file key. The keys are dynamically generated as needed and the approach is mostly to allocate one key to each communication session. For every new session, a new encryption key is generated, either by a common trusted party or by the communicators themselves. One good reason for doing this is to limit the amount of text to be encrypted under a specific key. Nevertheless, a session may be short or long depending on what type of communication is going on. This indicates that the amount of text enciphered under a single key can vary considerably. In other words, the "cryptoperiods" do not have similar length. As a result, the keys used in the same cryptosystem do not have a uniform security load.

There are various ways of handling this problem. Changing the key in the middle of a transmission can be carried out by introducing extra signalling, for example by adding another type of data packet. However, to modify the formats and functioning of an existing communication protocol is most likely not a viable solution. Another possibility is to modify the initial phase of connection. The protocol can negotiate a

J.J. Quisquater and J. Vandewalle (Eds.): Advances in Cryptology - EUROCRYPT '89, LNCS 434, pp. 110-120, 1990.

period length parameter during the set up procedure, which is then used throughout the entire connection.

This paper proposes to change the encryption key as a function of the bits communicated. If this can be carried out in a randomized manner, the cryptanalyst can now no longer be confident that all ciphertext intercepted from one session is encrypted with the same key. Choosing a suitable mode of operation, the only way to discover the cryptoperiods seems to be to break the specific encryption algorithm applied. By selecting an appropriate value of some parameter, the amount of text enciphered by each key can even be reduced to the unicity distance with some probability, if enough keybits can be spared, without reducing effective bandwidth of the channel.

A direct way to diffuse the duration of the cryptoperiods is to choose the lengths independently and according to some suitable probability distribution, either prior to, or during the transmission phase. An acceptable solution could be to select uniformly between a lower and an upper length limit. Such a method appears to be applicable to diverse classes of communication protocols that involve cryptographic functions where the secret keys must be changed in a randomized mode.

The special solution presented herein randomize the cryptoperiods based on the occurrence of patterns in a pseudo-random bit sequence. Moreover, I suggest to make use of a pseudo-random stream that is readily available in the cryptosystem itself; the cipher stream.

Some advantages of the scheme to be presented are:

- The exact time of key-change is kept secret (if the cryptoalgorithm is not broken); accordingly, no guarantee exists for a wiretapper that a single key is used for all ciphertext intercepted during a communication session.
- No extra signalling needed in the communication protocol,which makes it easy to add the technique to existing communication protocols.
- Data transfer is not interrupted regardless of session duration because key-change can be done at network speed.

2 Description of the technique

A strong cryptosystem can be characterized by many properties. Among them is the property of generating a strong pseudo-random bit sequence. Such a sequence should be indistinguishable from a random source of bits, in order to thwart statistical attacks of any kind. For the purpose of randomizing, it is sufficient to assume that the stream is indistinguishable from a random source with respect to the specific statistical test of matching patterns. However, for security reasons it must not be possible to distinguish between cipher streams encrypted under different keys.

The reliability and the authenticity of the bits received are taken for granted, for example by using the services of some link or transport protocol. Exploiting the pseudo-randomness of the cipher stream and its integrity, the cipher stream can be used to signal when the common event of key-change occurs. Both sender and receiver are scanning the cipherbit stream for a bit pattern or *flag* to appear, and when it appears the key will be replaced by some predetermined algorithm.

Neither the receiver nor any eavesdropper will know in advance when the next key-change is going to take place. However, the receiver will be able to detect the event by using some shared secret with the sender: the flag. The eavesdropper seems to be no better off than guessing the exact time of key-change, lacking information about which flag to look for and being unable to distinguish between streams enciphered under different keys.

If the receiver is able to distinguish between meaningful and meaningless decipherments, then he can decide with probability close to unity at which point the cryptoperiod terminates. If the next key to be used is known or can be computed at this stage, there is no need for explicit knowledge of the flag. The disadvantages of this method of decryption is the extra amount of computation and back tracking that has to be carried out whenever a period ends. The knowledge of the flag patterns make the decryption process simpler and faster for the legitimate receiver, and is accordingly assumed in the model.

If a reliable and authenticated multicast protocol can be provided, it is possible to extend the technique to include multiple receivers [Mjo89].

A special case will be to keep the flag pattern fixed. Now it becomes possible to resynchronize the encryption-decryption process at any time during a session, assuming a method for determining the new key is agreed upon in advance. Henceforth it is possible to relax the reliability demands of the channel. Bit synchronization can be obtained throughout the communication session with granularity as a function of the length of the flag.

2.1 The model

A set W contains all possible words of length l over the binary alphabet $\{0,1\}$. A set of words $\{w_i\}_{i=1}^{n}$ is picked at random from W. Similarly, a set of encryption keys $\{k_i\}_{i=1}^{n}$ is chosen at random from the set of possible keys K. The set of ordered pairs $\{(w_i, k_i)\}_{i=1}^{n}$ represents the shared secret between sender and receiver. A cipher stream is generated by an encryption algorithm $C = E_{\{k\}}(M)$ by input of a message stream M. Both sender and receiver carry out the following algorithm in order to switch to a new encryption key:

Init: Set $key \leftarrow k_1$; $flag \leftarrow w_1$; $i \leftarrow 1$
and start sending/receiving cipherbit stream C.

Matching: Scan stream C until match with *flag*.

Newperiod: Set $i \leftarrow i+1$; $flag \leftarrow w_i$; $key \leftarrow k_i$
and continue with "Matching".

This is a very high level algorithm description that hides much of the details. One concern might be traffic analysis of a possible delay in transmission caused by changing the keys. A fast matching detector and use of buffering can prevent this. More considerations for implementation are treated in section 4. The matching procedure itself can be performed differently. The details of this procedure affect the probabilistic model as shown in the next section. For this purpose it is useful to distinguish between *hopping* and *sliding*.

Hopping: Hopping is defined as increasing the position in the stream by l bits between every test for match. No bits in the stream are used in more than one comparison.

Sliding: Sliding consists of increasing the position by one before next comparison. Except for the case when $l = 1$, succeeding comparisons are dependent because $l - 1$ bits are common between the two comparisons.

A variation of the scanning procedure is to make a hop before the first comparison after a match has taken place, and then proceed as for normal scanning until next match. This is called *sliding with initialization.*

3 The statistics of matches

This section presents two probability models of the scanning algorithm; the first model is a slightly modified geometric distribution, the other does not fit any standard model, and is partly based on results by Blom and Thorburn [BT82].

3.1 Assumptions and definitions

If the analysis that follows shall be valid to the scheme proposed, a basic assumption about the stochastic properties of the cipher stream must be made. For a strong cipher it is reasonable to assume that it is not possible to apply any statistical test and be able to infer useful information about the cleartext from the ciphertext, other than that it appears to be random. For the present analysis a weaker assumption is sufficient, namely that the cipherstream is indistinguishable from a random stream with respect to one specific statistical test: The flag matching process.

Assumption 1 (Randomness) *The flag matching process cannot distinguish between C and a sequence of independent and identical distributed random variables.*

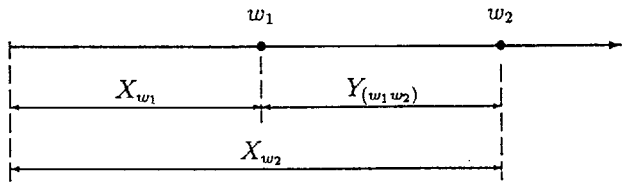
Let $C = [c_1, c_2, \cdots c_t, \cdots)$ be the semi-infinite sequence of cipher bits. A subsequence of length l at position t is denoted $[c_{t-l+1}, \cdots, c_{t-1}, c_t]$. A flag of length l is one of the 2^l different bit patterns possible.

Definition 1 (Match) *There exists a match between the flag w of length l and the sequence C at position t if $w = [c_{t-l+1}, \cdots, c_{t-1}, c_t]$.*

It follows from this definition that a match can only occur at position $t \geq l$. The distance between two matches at positions t and t' is $|t - t'|$. The distance normally considered is the distance between succeeding occurrences.

Definition 2 (First match) *A match at position t in C is the first match of flag w if there exists no match of w for positions $l \leq r < t$.*

The figure below depicts position of matches and distances to and between them.



Let X_w be the waiting time or distance to first match of w when starting without any preconditions. Let $Y_{(w_1 w_2)}$ be the distance between consecutive matches of w_1 and w_2, in that order.

Note that the flag length itself is included in the distance until first match. The distance between two matches is equal to the number of *new* bits examined.

In order to be consistent on the words "left" and "right", think of the scanning process as normal reading. All patterns are read from left to right. For an arbitrary pattern w, let the left r bits be denoted $w_{[r]}$, and the right r bits be denoted $w^{[r]}$. The notion of pattern overlapping is defined as:

Definition 3 (Overlap) *If the right r digits of w_1 are equal to the left r digits of w_2, then w_2* overlaps *w_1 with length r. This is denoted by the indicator variable:*

$$\varepsilon_r(w_1, w_2) = \begin{cases} 1 & \text{if } (w_1^{[r]} = w_{2[r]}) \\ 0 & \text{otherwise} \end{cases}$$

Note that the operator "overlaps" is not commutative in general. It is also convenient to define a notation for the total number of overlaps there are between two patterns:

$$\nu(w_1, w_2) = \sum_{r=1}^{l} \varepsilon_r(w_1, w_2)$$

If $\nu(w, w) > 1$ then w is *self-overlapping*. The longest overlapping subsequence between w_1 and w_2 is denoted $\kappa(w_1, w_2)$. If there exists no overlapping subsequence then $\kappa(w_1, w_2) = \lambda$, where λ is the empty string.

For convenience; $\varepsilon_r(w, w) = \varepsilon_r(w)$; $\nu(w, w) = \nu(w)$; $\kappa(w, w) = \kappa(w)$.

Example: If $\nu(w_1, w_2) = 1$, then $\kappa(w_1, w_2) = \lambda$. We have always that $\nu(w) \geq 1$ because $(w^{[r]} = w_{[r]})$ is always true for $r = l$.

3.2 Hopping

Recall that searching for the occurrence of a flag pattern in a random sequence by shifting l bits between successive comparisons is termed hopping, where l is the length of the flag. By assumption 1 and the fact that l new bits of the sequence are compared with the flag lead to independence between comparisons. The probability of getting a match in one comparison is $\Pr(\text{match}) = p = 2^{-l}$, by symmetry.

Let T be the number of comparisons until first match. From the conditions it follows that the random variable T is geometric distributed:

$$\Pr(T = t) = p(1-p)^{t-1}$$

where t is the number of comparisons carried out.

The position in the sequence after t samples will be lt, and hence the expected distance to first match is:

$$E(X_w) = l \cdot E(T) = l/p = l \cdot 2^l \tag{1}$$

where $E(T)$ is the expectation of the geometric distribution.

The variance of the distance in bits to the first match is

$$Var(X_w) = l^2 \cdot Var(T) = l^2(1-p)/p^2 = \frac{l^2(1-2^{-l})}{2^{-2l}} \tag{2}$$

where $Var(T)$ is the variance of the geometric distribution.

Let $Y_{(ww)}$ be the number of comparisons between matches of a specific flag w. Then the distribution of $Y_{(ww)}$ is equal to the distribution of X_w because the conditions are the same. Similarly, the distribution of the number of comparisons between *any* pair of flags of length l is equal to the distribution of X_w. Therefore the equations 1 and 2 also apply to the distance *between* matches.

Lemma 1 *For all* $w, w_1, w_2 \in \{0,1\}^l$: X_w *and* $Y_{(w_1 w_2)}$ *are identically distributed when the matching process is performed by hopping. The expectation of the distance between any pairs of flags is* $E(X_w) = E(Y_{(w_1 w_2)}) = l \cdot 2^l$.

In other words, the frequency of matches is a random variable with a distribution only dependent on the length of the flag. The choice of the length of the flag determines the expected frequency of key-changes to be $1/E(X_w)$.

Example: If the flag is 8 bits long, the expected distance between key-changes is 256 bytes (2048 bits). Increasing the length to 12 bits, the expected distance between key-changes becomes 6144 bytes (49152 bits). The standard deviation has approximately the same value as the expectation.

3.3 Sliding

Searching for the occurrence of a flag pattern in a random sequence by shifting only one bit between successive comparisons is termed sliding. Trivially, if the length of the pattern is only 1 bit, the stochastic conditions are still according to the geometric distribution. In general, however, interdependencies are introduced which complicates the model to some extent.

The expected distance between two matches of the same flag can be found by the following lemma:

Lemma 2 *For all* $w \in \{0,1\}^l$ *the expected distance between consecutive occurrences of* w *is* $E(Y_{(ww)}) = 2^l$.

The proof is based on the elementary renewal theorem. Note that there is no restrictions on the pattern of w.

The sliding procedure prescribes that the flag pattern shall be changed after each match. If this is neglected then the lemma can be used, but choosing another flag introduce dependencies, and the process seize to be a renewal process. These

dependencies can be formalized by the definition of overlapping sequences. And there is an indirect way of finding the probability distribution by looking at distances to first match.

Let the set of flags be restricted to those that are not self-overlapping $\{w_n\}$, then the following lemma connects the two distances X_{w_n} and $Y_{(w_n w_n)}$:

Lemma 3 *For all $w_n \in \{0,1\}^l$; if $\nu(w_n) = 1$ then the expected distance to first match is $E(X_{w_n}) = E(Y_{(w_n,w_n)}) = 2^l$.*

Proof(sketch): Let w_n be a flag for which $\nu(w_n) = 1$, and let X_{w_n} be the distance to first match of this flag. By observing that X_{w_n} and $Y_{(w_n,w_n)}$ are identically distributed because the flag is not self-overlapping, the conclusion of the lemma follows. □

It is now possible to generalize to self-overlapping flags where $\nu(w) > 1$.

Theorem 1 *Let X_w be the distance to first occurrence of the flag w. The expected distance to first match of $w \in \{0,1\}^l$ is*

$$E(X_w) = \sum_{r=1}^{l} \varepsilon_r(w) \cdot 2^r$$

Proof: The proof is based on applying lemma 2 and lemma 3.

The claim is that the distance to the first match of w is the sum of the distance to the first match of $\kappa(w)$ and the distance between successive matches of w:

$$X_w = X_{\kappa(w)} + Y_{(ww)} \tag{3}$$

where $X_\lambda = 0$.

Assume, without loss of generality, that the bitstring $\kappa(w)$ is at position t. Then the initial necessary and sufficient conditions for $Y_{(ww)}$ are exactly satisfied at position t. By recursively applying equation 3 $\nu(w)$ times, the distance X_w can be expressed as a sum like this:

$$X_w = Y_{(ww)} + Y_{(\kappa(w)\kappa(w))} + \ldots Y_{(\kappa^{\nu-1}(w)\kappa^{\nu-1}(w))} + X_{\kappa^{\nu-1}(w)}$$

where κ^r denotes the function product $\kappa \circ \kappa \cdots \kappa$ where κ appears r times. $\kappa^{\nu-1}$ represents the shortest overlapping subsequence of w and hence is not a self-overlapping sequence itself. By lemma 3 $X_{\kappa^{\nu-1}(w)} = Y_{(\kappa^{\nu-1}(w)\kappa^{\nu-1}(w))}$. Definition 3 implies that

$$\varepsilon_r(w) = 1 \Leftrightarrow r = |\kappa^s(w)|,\ s = 0 \ldots \nu(w) - 1$$

The theorem follows by using lemma 2 and asserting independence between the random variables. □

The next theorem gives the expected distance between flag w_1 and flag w_2:

Theorem 2 *For all $w_1, w_2 \in \{0,1\}^l$; the expected distance between consecutive matches in the order w_1 and w_2 is:*

$$\begin{aligned} E(Y_{(w_1 w_2)}) &= E(X_{w_2}) - E(X_{\kappa(w_1,w_2)}) \\ &= \textstyle\sum_{r=1}^{l} \varepsilon_r(w_2) 2^r - \sum_{r=1}^{|\kappa(w_1,w_2)|} \varepsilon_r(\kappa(w_1,w_2)) 2^r \end{aligned}$$

Proof: If $\kappa(w_1, w_2) = \lambda$ then $Y_{(w_1 w_2)} \geq l$, which is equivalent to theorem 1. More general, the lower bound on the distance is $Y_{(w_1 w_2)} \geq l - |\kappa(w_1, w_2)|$. Since $\kappa(w_1, w_2)$ is a subsequence of w_2, the distance to first match of w_2 is $X_{w_2} = X_{\kappa(w_1,w_2)} + Y_{(w_1 w_2)}$. By assumption 1 the variables $X_{\kappa(w_1,w_2)}$ and $Y_{(w_1 w_2)}$ are independent.□

Now it is possible to state the following bounds from theorem 1 and 2.

$$2^l \leq E(X_w) \leq 2^{l+1} - 2$$

$$2 \leq E(Y_{(w_1 w_2)}) \leq 2^{l+1} - 2$$

If the flag pairs (w_1, w_2) are chosen at random the average expected distance between matches turns nicely out to be $E(Y_{()}) = 2^l$ for long streams.

Remark: It is possible to find the probability generating function for the distance to first match [BTS2], and hence the probability distribution, the expectation and variance can be derived from that function [Mjo89].

Example: If the flag is 8 bits long, the expected distance between matches will vary from 2 bits to 510 bits. For 12 bits, the expected distance runs to 8190 bits, for 16 bits length the expected distance range 131070 bits.

3.4 Deciding cryptoperiods

The main classes of attacks (ciphertext-only, known-plaintext and chosen-plaintext) assume normally that the same key is used for all text involved. Changing keys more often at random places can make these attacks harder to achieve.

Keeping the flag constant gives 2^l possible partitions of a given cipher stream. This is independent of the length of the stream, so that it is not hard to test exhaustively for reasonable lengths of the flag.

Changing the flag by every match makes the number of possible partitions grow exponentially with the length of the cipher stream. Let $1/\delta$ be the matching intensity, and $|C|$ the length of the communication session, then the number of possible partitions is $O((2^l)^{|C|/\delta})$.

Selecting flags independent insures that detecting one cryptoperiod does not reveal anything about past or future events.

3.5 A simple cryptanalytic model

One possible start for a cryptanalytic attack is to include the stack of keys and flags as part of the cryptoalgorithm, and then go on and analyze the box as an autonomous bit generator. A basic algorithm might be a linear feedback shiftregister enhanced with a stack of states (the keys) and a bit pattern detector. This also indicates a possible method of making a linear generator non-linear, although a first analysis shows that quite strong restrictions on the feedback must be assumed [Ing88].

4 Implementation considerations

In practice a decision has to be made about when exactly the key change shall occur, which is highly dependent on the specific cryptosystem applied. Looking

for a specific pattern match by scanning the cipher stream is easily implemented by hardware containing a register and some comparator circuitry. This allows for minimal delays such that the output stream will not be interrupted because of a key change.

A lower and an upper bound on the number of bits encrypted with the same key can easily be enforced with a counter, in order to avoid those rare occasions where a match does not take place after a reasonable number of bits.

Of course, it is of no use to set the expected distance between matches shorter than the blocksize of the cryptosystem. The clustering of events is still a problem that need careful attention. A possible solution is to use sliding with initialization, or even to skip the rest of the block. The scanning process will then be according to theorem 2 plus the amount of bits that are skipped.

4.0.1 Variations and extensions

Some ideas for extensions and variations are only briefly mentioned here:

- Pseudorandom generation of flags and keys by some strong cryptographic function.
- Computing the next flag to be used by some one-way function of variables like a masterkey, time, input and output traffic and other state information.
- Scanning for the r'th match.
- Scanning the stream for several flags combined by a Boolean function.
- Keeping a pool of keys, picking the next one to use according to some one-way function of where the flag appears.
- Adding memory to the flag detector.
- Multicast protocols.

5 Related work

Nicolai[Nic83] suggested a method of increasing the the linear complexity of a key bit generator by making random jumps in the cycle of the generator. He exemplified this by describing a generator constructed of several maximal length linear feedback shiftregister sub-generators that are XORed together. It is possible to jump from one point in the cycle to another by modifying the state of some of the subgenerators, for example by having the ability to step each sub-generator independently. The "code branch" should have the following properties:

1. The jump should be long to avoid reuse of key bits.
2. The jump is determined by a one-way function of the previous state of the generator.

The receiver can determine when a code branch has occurred by detecting error in decipherment, for example by relying on some integrity check. When this happens the decryption procedure backtracks until a correct decipherment appears.

Applying the scheme proposed in this paper, the code branching event will occur when the flag matches the ciphertext. If the flag patterns are chosen independent and at random then the code branching will be randomized. The backtracking procedure is not necessary if the flag sequence is predistributed. Still there is the need for some integrity check in order to keep synchronization. Using a one-way function of the previous state to determine the target state of the jump appears to a practical solution of key generation.

During the reviewing of this paper another related reference was brought to my attention. A Swiss patent of 1986 [KW86] describes a similar scheme, for the purpose of maintaining cryptographic synchronism between transmitter and receiver. The patent addresses the problem of cryptographic resynchronization in a stream cipher system where temporary errors such as bit slips are acceptable to a certain degree. Resynchronization is made possible by letting the flag pattern be fixed in advance. Whenever the flag appears in the cipherstream the key-bit generator is initialized as a function of a given number of bits following the predetermined bit pattern. This method is currently in widespread use [Sch89].

The fixed flag scheme appears to be the converse of what is presented in this paper. While the patent claims that diffuse cryptoperiods can be made distinct by flag matching, this paper focus on using flag matching to obtain diffuse cryptoperiods. Moreover, it is crucial that the flag pattern is changed continuously.

6 Generalizing the problem

The original idea was to diffuse the cryptoperiods of a cipher stream. A cryptoperiod can loosely be defined as that part of a message stream enciphered with the same key. However, synchronizing change of encryption keys is a special instance of any common event between sender and receiver. The only necessary property of the event is that it can be related to the bitstream in some unique way. The target of the action of the event need not be the same stream, like the cryptoperiod diffusion scheme presented here.

If the consequence of the event is purely internal on both sides, there is no method to detect this from the outside. In the special case of switching encryption keys, this can in principle be computed from the outside given sufficient resources and ciphertext. We may define an internal action *inobservable* if what is communicated to the outside is independent of the change of internal state. Following this, an internal action is *computationally inobservable* if it is infeasible by any computation to detect the change of state, even though sufficient information is released to the outside. A synchronized internal action is originated by an event synchronized between two or more parties. An example of a synchronized, computationally inobservable action is the key-switch proposal of this paper.

Matching flags can be seen as a very specific and limited statistical test of the stochastic properties of the cipher stream. Any suitable test function, with or without memory, might be used instead.

Acknowledgements

My thanks to Ingemar Ingemarsson for commenting on the main idea and adding another; To David Chaum for pointing out related work and reading of an early manuscript; To Bo Lindquist for discussion and finding a useful reference in the literature; And to Bjarne Helvik for numerous helpful discussions.

References

[BT82] G. Blom and D. Thorburn. How many random digits are required until given sequences are obtained? *J. Appl. Prob.*, 19():518–531, 1982.

[DP84] D. W. Davies and W. L. Price. *Security for Computer Networks*, chapter six. Wiley, 1984.

[Ing88] Ingemar Ingemarsson. Bitmönsterstyrda händelser. November 1988. Private communication.

[KW86] H. Klemenz and W. R. Widmer. Verfahren und vorrichtung zur chiffrierten datenübermittlung. Swiss patent CH 658 759 A5, November 1986.

[Mjo89] S. F. Mjølsnes. *Some Issues in Cryptographic Protocols.* PhD thesis, The Norwegian Institute of Technology, Trondheim, 1989. In preparation.

[MM82] Carl H. Meyer and Stephen M. Matyas. *Cryptography: A New Dimension in Computer Data Security*, chapter six and seven. Wiley, 1982.

[Nic83] Carl R. Nicolai. Nondeterministic cryptography. In D. Chaum, R. L. Rivest, and A. T. Sherman, editors, *Advances in Cryptology: Proceedings of Crypto '82*, pages 323–326, Plenum Press, New York, 1983.

[NS78] R.M. Needham and M. Schroeder. Using encryption for authentication in large networks of computers. *Communications of the ACM*, 21(12):993–999, December 1978.

[Sch89] P. Schmid. Private communication. June 1989. Omnisec AG, Regensdorf, Switzerland.