Cryptographic Authentication of Responses in ATM Traffic Control Protocols

Werner Almesberger
werner.almesberger@lrc.di.epfl.ch

Laboratoire de Réseaux de Communication (LRC)
EPFL, CH-1015 Lausanne, Switzerland

February 28, 1996

Abstract

The control flow of ATM traffic control protocols frequently includes untrusted elements (e.g. end systems), but there is no proper validation of their responses. This paper discusses the problem, describes possible solutions, and proposes the use of a cryptographic authentication scheme with very good scaling properties. Requirements for the cryptographic algorithm are specified, applicability of that concept to several usage scenarios is examined, and necessary architectural extensions to the traffic control protocol are illustrated on the example of ABT/VT.

1 Introduction

Many traffic control protocols for ATM include untrusted elements such as "private" networks or end systems in the flow of control information by having the network send them RM cells [1] containing requests or acknowledgements which the end systems are expected to return to the network afterwards. In the following, we will focus mainly on end systems, but the concept applies equally to untrusted branches of the network. The end system is typically expected to either return such cells unaltered or to perform a restricted set of operations on them, e.g. to reduce the requested resource allocation.

This can be illustrated on the example of the ABT/VT [2] protocol, where the source emits reservation requests, which are forwarded hop by hop to the destination. The network and the destination may reduce the resources requested. The destination returns the reservation requests (they are now called "reservation acknowledgements") through the network back to the source and finally, the source sends them again, this time to claim the reservation that has been acknowledged by the network and by the destination. See also figure 1.

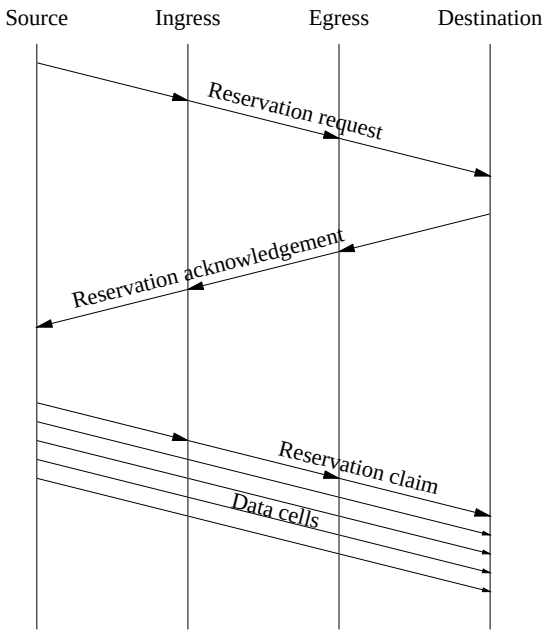

Figure 1: Message flow in ABT/VT

The ABT/VT terminology (namely "request", "acknowledgement", and "claim") will be used throughout this paper for the corresponding semantics. This does, however, not restrict applicability of the concepts discussed herein to that particular protocol.

Potential dangers

Protocols that include end systems in the control flow have the problem that, while the network is usually considered trustworthy, the end system or possibly even two (or, for multicast, even more) cooperating end systems may, be it by accident or intentionally, send acknowledgements or claims that exceed the resources that have previously been granted by the network. If this goes undetected, it may result in unfairness or worse.

There are several ways to resolve this problem, e.g.:

- keep control flows inside the network, possibly adding a local control flow for the end system
- examine all responses from end systems for validity
- base traffic policing solely on what has been granted by a trustworthy party
- the most common approach: ignore it

While the use of several control flows can be important for the wide area [3], it adds more complexity than would be desirable for protection against misbehaving end systems. Basing traffic policing only on trusted information may be as difficult as validating all end system responses, see below.[1] Simply ignoring the problem may be a very reasonable approach for networks where a certain amount of cooperation and tolerance can be expected. Unfortunately, this is not applicable to cases where "hard" service guarantees are required.

Validation of responses

End system responses can be validated either by verifying general properties (e.g. if they exceed the bandwidth that has been granted for the last few requests), or by validating each individual message. "Generalization" tends to become less accurate or to require more hardware when more requests are in transit because round-trip time increases. This can be avoided by only allowing a limited number of messages to be present in the network at the same time (e.g. ABT/DT [4], [1]), but this is not universally applicable. Validating based on remembering exactly what has been sent and by looking up incoming responses has scalability problems too, i.e. it requires storing information for the duration of one round-trip between the point where the validation is performed and the end system.[2]

We propose the use of cryptographic methods to ensure validity of end system responses. The network itself it used to "store" the information needed to perform the validation, so the system that performs the validation does not incur the scalability problems associated with other approaches.

2 Operation

The concept of using cryptographic means to ensure integrity and authenticity of messages is well-established for higher protocol layers (e.g. [5], [6], [7]). Recently, the use of cryptographic authentication has also become increasingly popular for comparably short-lived data, e.g. IP datagrams [8].

Concept

The general concept is to validate end system responses (R) based on the content of the original message (O) they received. This is done by using a function that determines if $R \in M$, with M being the set of possible modified values the end system may return in response to O. The part of O relevant for validation is carried in R and is protected against tampering by a message authentication code (MAC; i.e. a secure keyed one-way hash).[3]

For authentication, a secret key (i.e. unknown to the end system) is used to modify the result of

1. Note that protocols like ABT/VT simplify traffic policing by using RM cells to announce traffic changes.

2. But it may be good enough if the number of outstanding responses is always very small, e.g. in LANs.

3. More sophisticated approaches where the encryption algorithm itself helps to confine possible manipulations (e.g. along the lines of the algorithm used by S/KEY [6]) might deserve further consideration.

the hash function.[4] If the end system must be able to modify the data passed in the message, it can store the modified data in an unprotected message region and the network can then verify validity of the modification and copy the modified data into the trusted section of the message. Figure 2 illustrates the general message structure for O and R.

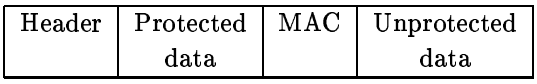

Figure 2: General structure of RM cells with a message authentication code

Storing the protected data at the beginning of the message gives the validation function more time for the accept/reject decision.

If an inconsistency is found, the corresponding information is ignored and flagged as invalid or absent. The cell is discarded if it carries no other relevant information.

Algorithms

The algorithm given below illustrates the procedure. `C` is the cell being processed, `C.M` is the MAC, `C.P` is the protected data area, `C.U` is the unprotected data area, and `S` is the secret key. `hash()` is the secure hash function, `mod()` is the local function used to modify (e.g. degrade) a request, `check()` is the function that decides whether its second argument is a valid modification of its first argument, and `send()` is the function that performs normal sending or forwarding of the RM cell.

The system performing the validation executes the following operations when sending towards the end system:

```
C.P = mod(C.P);
C.M = hash(S,C.P);
send(C);
```

When receiving from the end system:

```
if (hash(S,C.P) != C.M) error;
if (!check(C.P,C.U)) error;
send(C);
```

And the end system simply does:

```
C.U = mod(C.P);
send(C);
```

Attacks

Any such authentication method has to be robust against:

- replay attacks
- use of messages in an incorrect context
- attempts to obtain the secret key
- attempts to use an equivalent key
- attempts to guess a valid MAC

Replay attacks can be avoided by using a strictly monotonously increasing sequence number. The sequence number has to be part of the protected data. To limit the sequence number space (e.g. to 32 bits), sequence numbers can be reused if the secret key is changed at least whenever the sequence number wraps. (See below for how to change keys.)

A similar type of attack is to use contents of a message not pertaining to the same connection, e.g. a message (possibly of a different type) received on a different VC or even a message received by a different host. In order to protect against that, a different key should be used for each connection. If breaking the secret key can be considered as hard, it would be sufficient to generate one "good" (i.e. sufficiently random) key per end system or even per switch, and to include the connection identifier (VPI and VCI) in the MAC.

The key is protected by the strength of the hash function. Also, the probability of finding a different key that results in the same MAC depends mainly on the hash function. Besides choosing a reasonably strong hash function,[5] the protection can be improved by regularly changing the secret key. The key change can be synchronized by using the sequence number, which therefore should be stored at the beginning of the protected data. Key changes should be rare enough that it would be sufficient to use only two keys (old key and new key) at a time. Furthermore, the key must be difficult to guess (see also [9]).

4. Note that, given that the key may have to be chosen based on the sequence number (see below), using a constant initial seed and hashing the secret key after the sequence number has been received may yield a efficient implementation.

5. Note that the hash function also has to resist chosen plaintext attacks.

3 Usage scenarios

The concept presented above is mainly suitable for scenarios where a branch of the network (e.g. a private network attached to a public network) is not trusted by the rest of the network, but nodes belonging to that branch, including the end system, trust each other, see figure 3. We call the flow of control information from the system performing the validation to the end system and back to the validating system a "protected loop".

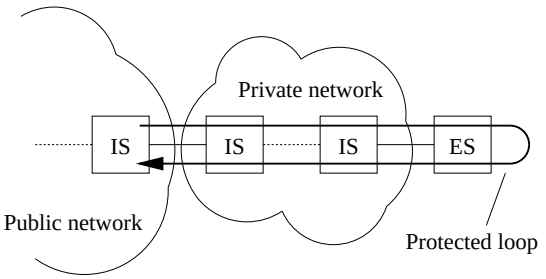

Figure 3: Example of the generalized scenario

Nested loops

The use of a second protected loop, e.g. between the private network and the end system (see figure 4) appears to be a natural and desirable extension of the concept.

If both, the system at the outer loop and the system at the inner loop need to be able to detect invalid responses, space for the MAC and for the protected data has to exist for each loop.

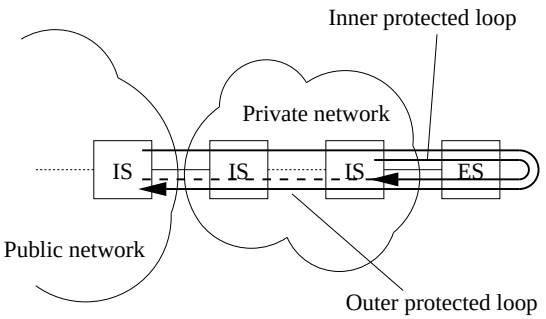

Figure 4: Double protected loop

If it is sufficient to perform the actual validation only at a single point (at the system the farthest away from the end system), only copying of the part of protected data that can be changed, and merging of both MACs (see below) is required.[6] The dashed line in figure 4 indicates how the detection of invalid responses is delayed by this simplification.

Each additional level of protection requires copying of the same amount of data, thereby making it unrealistic to support more than a very small number of levels. For completeness, an algorithm for an arbitrary number of levels is given below.

Note that the cryptographic approach can of course be mixed with other approaches, e.g. a small table for verifying responses may be sufficient at a switch talking to a workstation, and the more powerful cryptographic authentication could be used only at a network-network boundary.

Algorithm for nested loops

The following algorithm handles an arbitrary number of protection levels. The system performing validation on the outmost loop is expected to do all the error checking for the other systems. In addition to the fields and functions already defined, the copy area `C.F` is added, which is organized as a LIFO with the operations `push()` and `pop()` to add or to remove elements, respectively. Signaling is assumed to handle limiting of the number of levels.

The system at the outmost loop and the end system use the same algorithm as for the single-loop case. Depending on the implementation, the system at the outmost loop may have to initialize `F`.

Each system that adds validation for an inner loop uses the following algorithm when sending towards the end system:

```
push(C.F,C.P);
C.P = mod(C.P);
C.M ^= hash(S,C.P);
send(C);
```

When receiving from the end system:

6. Note that this protection is only good enough to prevent attempts to obtain a better service than allowed, but that the inner system is still susceptible to denial of service attacks.

```
C.M ^= hash(S,C.P)
if (!check(C.P,C.U)) error;
C.P = pop(C.F)
send(C);
```

4 The secure hash function

The secure hash function must fulfill the following requirements:

- it must be sufficiently "hard" to resist attempts to break it during the life time of a key (in particular, chosen plaintext attacks need to be considered)
- its properties must be well understood in order to be reasonably sure that no simple attacks will be found in the near future
- it must be quickly computable; ideally it would process data at the rate at which it arrives
- its implementation must not be inherently expensive

MAC and secret key size

The MAC size must only be sufficient to protect against successful "guessing" of a valid MAC.[7] The protection can be strengthened by monitoring repeated failure to guess the MAC and by simply disconnecting the misbehaving party.

The secret key must be sufficiently large to be unlikely to be broken within a key change interval.

Exact values depend on environmental parameters (e.g. desired strength of protection, attack frequency, number of attackers).

Choosing an algorithm

Most of the secure hashing algorithms given in standard cryptography literature [12] are cryptographically strong but even the fastest ones tend to be comparably expensive to calculate, e.g. the MD5 algorithm [10] (see also [11]).

The use of cryptographically weaker algorithms should be studied for cases where stronger algorithms are too complex or too slow. Combined with frequent key changes (e.g. once every few seconds[8]) and the use of independently generated keys for each connection, even a weak protection may be sufficient.

5 Case study

The changes in message formats when using MACs for traffic control response validation is studied in this section on the example of the ABT/VT protocol. The same concepts can be readily applied to similar protocols, such as ABR [3] or ABT/DT[9].

Message format

ABT/VT uses three messages with the following fields:

- reservation request: requested cell rate (RCR) and requested block size (RBS)
- reservation acknowledgement: reserved cell rate (RCR) and reserved block size (RBS)[10]
- reservation claim: claimed cell rate (CCR) and claimed block size (CBS)

Furthermore, a reservation request and a reservation claim can be merged into the same message.

Protection is necessary in two cases: (1) when the destination turns a request into an acknowledgement, and (2) when the source turns an acknowledgement into a claim.

Because it is desirable to store protected data near the beginning of the cell, we use fixed locations based on their protection, and the effective location of **R** (RCR and RBS) and **C** (CCR and CBS) varies depending on the security needs in a given context. In addition to the actual traffic control information, we also need a message type byte, the sequence number, the MAC (e.g. MD5 folded into 8 bytes), and a CRC. Figure 5 shows the complete cell payload.

7. The MAC should be as small as possible, given that typically only 48 bytes are available to store header, traffic control information, sequence number, MAC, and usually also a CRC.

8. If only one pending key change is allowed at a time, the key can be changed after at least one round-trip (along the protected loop) plus the maximum queuing and processing delays.

9. In cases 3 and 4 explained in annex C of [1]

10. Those fields have the same name in request and acknowledgement messages, because they're actually the same variable: a *reserved* entity is just a *requested* entity after all involved parties have accepted or degraded the request.

12. Note that [2] does not specify the exact cell format, so "worst case" sizes have been chosen for some fields.

Protocol identifier	1 byte
Message type	1 byte
Sequence number	4 bytes
Trusted cell rate	4 bytes
Trusted block size	4 bytes
MAC	8 bytes
Untrusted cell rate	4 bytes
Untrusted block size	4 bytes
Reserved	16 bytes
CRC-10	2 bytes

Figure 5: RM cell payload for ABT/VT with a MAC[12]

Processing

Figure 6 illustrates the processing necessary to verify correct end system behaviour in the two cases described above. Each message contains the following fields (from left to right): trusted data, the MAC, and untrusted data. The trusted data field may contain untrusted data if there is no MAC. Field contents are labeled as follows: R is a request block, R′ is a modified request block (i.e. an acknowledgement), C is a claim block, and X, Y, and Z are MACs.

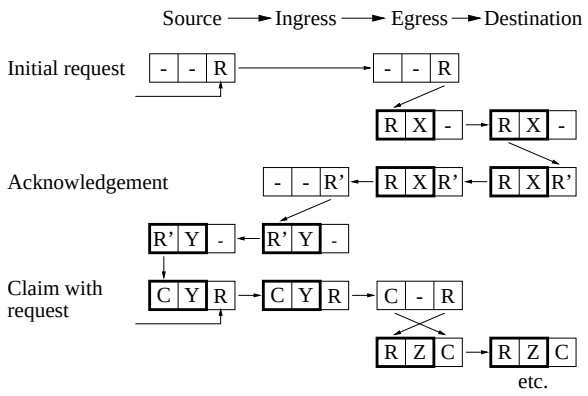

Figure 6: Authentication processing for ABT/VT

The source emits the first reservation request, containing only the request block. That request propagates through the network, where it may be degraded at each switch. The egress switch copies the request block to the trusted data area, assigns the sequence number,[13] computes the MAC X, and sends the reservation request to the destination.

The destination copies the requested cell rate into the untrusted data area, yielding R′. It may also degrade the request at the same time. The resulting reservation acknowledgement is sent to the egress switch, where the validity of X and R′ is checked, i.e. that $R' \leq R$. If the message passes the check, it is forwarded through the network to the ingress switch. Note that the values of the trusted data area and of the MAC only have a meaning to the egress switch and become therefore irrelevant when the message is sent towards the ingress switch. The ingress switch copies the reservation information to the trusted data area, computes the MAC Y, and sends the reservation acknowledgement to the source.

When the source decides to issue the reservation claim, it adds a new request and sends the combined request and claim message to the ingress node. The name change from R′ to C only reflects the difference in meaning.[14] The ingress switch verifies the MAC, optionally degrades the request part, and sends the message to the egress node. Again, the MAC becomes useless after the message has left the ingress switch. The egress switch swaps the trusted and the untrusted data areas, computes the MAC Z, and forwards the message to the destination, which then continues as described above.

6 Conclusion

The need to validate responses from end systems which are part of the control loop of ATM traffic control protocols has been identified and we propose a scalable method that is based on cryptographic message authentication.

The mechanisms necessary to implement such validation have been described at the conceptual level and various aspects for choosing the secure hash function, the MAC size, and the number of random bits in the secret key have been discussed.

13. If multiple nested loops are allowed, the sequence number should be generated by the source or by the ingress switch. For simplicity, we assume in this example that each loop is independent of all the others.

14. ABT/VT allows no modification of the reservation at that point.

As an illustration, the use of MD5-based authentication for ABT/VT has been described.

This paper does not attempt to give a concluding statement on the choice of the secure hash function or the MAC size, but suggests further research, particularly in the area of algorithms that are considered "too weak" for classical cryptography applications.

7 Acknowledgements

The author thanks Germano Caronni for very helpful discussions on contemporary cryptography, and Jean-Yves Le Boudec for valuable advice and suggestions.

References

[1] ITU-T Recommendation I.371. *Traffic control and congestion control in B-ISDN* (Perth draft), ITU, November 1995.

[2] Almesberger, W.; Le Boudec, J.-Y.; Manthorpe, S. *An ABT based ABR service for ATM*, Technical Report *to appear*, EPFL, February 1996.

[3] The ATM Forum, Traffic Management Working Group. *ATM Forum Traffic Management Specification Version 4.0*, ATM Forum contribution 95-0013R10, 1996.

[4] Boyer, P.; Tranchier, D. *A Reservation Principle with Applications to the ATM Traffic Control*, Computer Networks and ISDN Systems, vol. 24, pp. 321-334, 1992.

[5] Zimmermann, P. *The official PGP user's guide*, MIT Press, 1995.

[6] Haller, N. M. *The S/KEY One-Time Password System*, `ftp://ftp.bellcore.com/pub/nmh/docs/ISOC.symp.ps`, Bellcore, October 1993.

[7] Kohl, J.; Neuman, C. *The Kerberos Network Authentication Service (V5)*, RFC1510, September 1993.

[8] Atkinson, R. *Security Architecture for the Internet Protocol*, RFC1825, August 1995.

[9] Eastlake, D.; Crocker, S.; Schiller, J. *Randomness Recommendations for Security*, RFC1750, December 1994.

[10] Rivest, R. *The MD5 Message-Digest Algorithm*, RFC1321, April 1992.

[11] Touch, J. *Report on MD5 Performance*, RFC1810, June 1995.

[12] Schneier, B. *Applied cryptography*, Wiley, 1996.