APPROXIMATE MESSAGE AUTHENTICATION CODES

Richard F. Graveman
Bellcore
rfg@bellcore.com

Kevin E. Fu
Bellcore and MIT
fubob@mit.edu

ABSTRACT

A strong method to authenticate information in spite of small changes is described, analyzed, and implemented.

PROBLEM DESCRIPTION

In many communications protocols, authenticating the source and integrity of information is equally as important as preserving confidentiality. Even when confidentiality is the primary security concern, an analysis of data encryption protocols reveals that blindly accepting encrypted data as legitimate may endanger confidentiality [Bel96]. Message authentication codes (MACs) are a widely used method for Alice and Bob, who share a secret key, to ensure each other that their messages are authentic and unmodified [Sim91]. The creator of a MAC takes a message and key as inputs and computes a checksum in a way that is hard, given some set of messages and their MACs but not the key, to forge a MAC for a new message or to discover the key. The usual tools for constructing MACs are cryptographic hash functions [PGV93, Pie93, FIPS95] and block ciphers [FIPS93, FIPS80, and FIPS85]. However, all of these are deliberately constructed to be as unforgiving as possible. A change of a single bit of the message is designed to affect the calculation of all of the checksum bits and to change each one in roughly half of the cases.

In some applications (e.g., electronic payments) Alice and Bob's security requirement is to reject any message that has been altered to the slightest extent. In other applications (e.g., voice or imagery), incidental noise or the effects of lossy downstream compression are at least somewhat acceptable, so long as Alice and Bob can identify and reject all-out forgeries, substantial modifications of content, "cut-and-paste" attacks, etc. In certain applications, in particular the one that motivated this work [Arc97], the "acceptable modification" to the message may include the insertion of hidden data: e.g., digital watermarks or fingerprints to signify ownership or to mark a particular copy (see [ABGM99]). Three traditional approaches to this problem come to mind:

1. Use error correcting codes to restore the exact message. While often practical in the case of channel noise, this approach expands the data stream and runs counter to the purpose of lossy compression. It is for the most part independent of the goals of information hiding schemes, although some information hiding schemes use error correcting codes to increase robustness *within the hidden data.*
2. Pre-compute the expected modifications of the lossy compression scheme and compute a MAC based on the result of this expected operation. This may be practical in the cases of downstream compression, but it cannot anticipate unpredictable channel noise and can only handle hidden data that are already known at the time the message authenticator is created.
3. Compute a MAC on the statistics of the message that must be preserved (e.g., the high contrast edges in an image). The difficulty with this approach is capturing the essential properties and recording the correct granularity in the statistics, because these properties may vary from message to message.

A fourth approach is taken here. A new cryptographic primitive called an Approximate MAC (AMAC) is introduced. It is a probabilistic cryptographic checksum, based on a shared key, with the following properties:

1. The messages close to a given message (measured by Hamming distance) may all have the same AMAC (for example, if the messages differ in less than one bit out of every ten thousand).
2. A slightly larger difference between two messages tends to result in a small difference (i.e., small Hamming distance) between their AMACs (for example, if one bit out of every several thousand is changed, on the average).
3. The actual bit positions in which these small differences occur may or may not provide some information about the differences between the two messages. (E.g., some bit positions within a pixel or some coefficients in a transform domain may be more significant.) Thus, conveying information in the AMAC about the underlying data structures may be desirable in some applications.
4. Any change in the key has the same type of effect on AMACs as on ordinary MACs: each bit changes in approximately half the cases. The same is true

*Prepared through collaborative participation in the Advanced Telecommunications & Information Distribution Research Program (ATIRP) Consortium sponsored by the U.S. Army Research Laboratory under Cooperative Agreement DAAL01-96-2-0002. The U.S. Government is authorized to reproduce and distribute reprints for government purposes notwithstanding any copyright notation thereon. The views and conclusions contained in this document are those of the authors and should not be interpreted as representing the official policies, either expressed or implied, of the Army Research Laboratory or the U.S. Government.

for any change in the probabilistic "coin tosses" used to create a given AMAC.

AMACs do not directly address whether, for example, one source image was obtained from another by cropping, stretching, rotating, and so forth. There is no attempt made at pattern matching, so, for example, dropping bits through loss of synchronization, moving the top row of pixels in an image to the bottom, or deleting a column of pixels results in something entirely new, not in something "approximately the same." The security goals for AMACs address only authentication and approximate bit-by-bit data integrity. They aim simply (1) to alert one to the occurrence of small changes possibly resulting from unpredictable noise, insertion of watermarks or fingerprints, or lossy compression followed by decompression, and (2) to make all-out forgery and significant undetected tampering difficult. Digital watermarks or fingerprints themselves may, in turn, be required to be robust in the face of various attempts to remove, to disguise, or to alter them, but that is a separate question (see [ABGM99]).

It is unclear how to build an AMAC from traditional cryptographic primitives like hash functions and block ciphers, so an alternative was sought. Anderson and Manifavas [AM97] solved the following problem, which helped inspire the construction of this AMAC:

A movie will be broadcast in encrypted form to prevent theft of the broadcast signal, but each paying customer will be given a different decryption key, which will result in small differences in the decrypted data from customer to customer, so the broadcaster can identify particular decrypted copies.

AN EXAMPLE

Suppose that Alice's message, M, is a gray-scale image consisting of 1024 x 1024 eight-bit pixels, which amount to 2^{23} bits (one megabyte) altogether, and Alice will generate a 128-bit AMAC. Consider her message to be a table, M, of 2^{16} rows of 128-bit words, $M[0]$ through $M[2^{16} - 1]$, where the 2^{10} pixels (2^{13} bits) in the i^{th} row of the image occupy rows $64i$ through $64i+63$ of M. Suppose that Alice and Bob have:

- A shared key, e.g., computed by using Diffie-Hellman key exchange [DvOW92]. The length of the shared key is variable, but should be sufficient to prevent forgeries by trying all keys. In most of the anticipated applications, Alice and Bob will authenticate each other, generate a shared key, use it immediately, and discard it, so attacks against a given key are only possible "on line."
- A cryptographically strong pseudo-random number generator. In practice, this primitive can be the key stream generator for a stream cipher (e.g., RC4 [Sch96] or VRA [ARV99]). This PRNG will be called upon to produce about as many bits as the total sizes of the messages authenticated.
- A family of pseudo-random permutations. A block cipher like DES [FIPS93] or IDEA [Lai92] can be used. The present example uses 12 round, 16 bit RC5 [Riv95]. One such unpredictable permutation is selected and used to construct each AMAC.

For a given key, K, and message, M, Alice can produce a family of AMACs parameterized by an initialization vector (IV), I, which does not need to be secret. Each I defines different neighborhoods, that is, a different partition of messages into ones with the same AMAC.

The following specifies an algorithm for the construction of an AMAC A for a one-megabyte image M:

1. Alice and Bob agree on a shared key K. Then Alice performs steps 2 .. 15.
2. Choose an IV, I.
3. Seed the PRNG with K and I.
4. Use the PRNG to select a pseudo-random permutation P on 16 bit numbers. (For example choose a 96-bit key for 12-round, 16-bit RC5, which is then used to encrypt the entire range of values $0 .. 2^{16} - 1$.)
5. For $j = 0$ through 255, repeat steps 6 .. 10.
6. For $i = 0 .. 255$, repeat step 7.
7. Fill the array S with 256 128-bit words from M: Set $S[i] = M[P(256j + i)]$.
8. Set $S[256] = 0$.
9. Exclusive OR all 257 rows of S with pseudo-random bits from the PRNG.
10. For $k = 0 .. 127$, set the kth bit of $T[j]$ to the MAJORITY of the kth column of bits of S.
11. Set $T[256] = 0$.
12. Exclusive OR all 257 rows of T with pseudo-random bits from the PRNG.
13. For $k = 0 .. 127$, set the kth bit of A to the MAJORITY of the kth column of bits of T.
14. Send I and A to Bob over a reliable channel, e.g., using an error correcting code or an out-of-band channel.
15. Send M to Bob over the "noisy channel."

The idea behind the construction is as follows: permute the 2^{16} rows of M, and then regard the permuted M as 2^8 "pages" of 2^8 rows each. Then "encrypt" the permuted M by exclusive ORing M with random bits. Now each permuted, encrypted page of M consists of 256 rows and 128 columns. For each of these 256 pages, add a 257th row of bits from the PRNG to make the number of rows odd, and then build a new row by calculating the MAJORITY of each column. Make a new page called T out of these 256 MAJORITY rows, add a row of pseudo-random bits to give T an odd number of rows, and compute the MAJORITY of each column of T. These 128 bits, together with I, are the AMAC. Note:

- Bob receives a modified version of M, say M^*, performs the same calculations as Alice (steps 3 through 13), and gets A^*. If $(A \oplus A^*)$ has sufficiently low Hamming weight, he accepts M.
- If correlation of positions within a row is not desired, add step 7a after step 7:

7a. Obtain a seven bit number h from the PRNG and shift S circularly h bits to the left.

ANALYSIS

We assume that Alice and Bob compute A and A^* as above on slightly different versions of a 1 megabyte (2^{23} bit) file, and we calculate how much their AMACs will differ. If between 10^{-4} and 10^{-3} of the bits delivered to Bob from Alice are changed, he will see about 2^{10} to 2^{13} differences among the 2^{23} bits sent by Alice. These are distributed across 128 columns in the 256 pages (Bob's S-arrays) and may or may not affect the MAJORITY. We estimated the expected Hamming weight of $A \oplus A^*$ for 2^{10}, 2^{11}, 2^{12}, and 2^{13} bit changes as follows:

1. Compute the distribution function for the number of differences d per column across all $2^7 \cdot 2^8 = 2^{15}$ columns in the 256 pages of S-arrays[1] with a hyper-geometric distribution [Fel68] (Table 1).
2. Compute the distribution of the Hamming weights w of the columns of S (or T), which are balanced, independent, identically distributed bits (Table 2).[2]
3. Using (2), compute the probability that d differences in a given column will change the MAJORITY function (Table 3).
4. Using (1) and (3), compute the expected number of differences between the two T-arrays (Table 4).
5. Apply Bose-Einstein statistics to estimate the distribution of these differences across the columns of the T-array (Table 5) and repeat step (3) (Table 6).

[1] This calculation was repeated by considering the differences to fall into a hypergeometric distribution across the 256 pages and then approximating the distribution of these differences with Maxwell-Boltzmann and Bose-Einstein statistics [Fel68]. The results are slightly smaller this way, due to the "sampling with replacement" effect. For distance 4096, the expected value was 4.79 versus 4.82; for 8192, 6.75 versus 7.07.

[2] The MAJORITY of balanced i.i.d. bits is balanced i.i.d.

In summary, if M and M^* have Hamming distance 1024, $A \oplus A^*$ has an expected Hamming weight of 1.88 bits; distance 2048 produces an expected weight of 3.11 bits; 4096 yields an expected 4.82 bits; an 8192 bit difference results in an expected 7.07 bit weight.

	Hamming Distance between 1 Megabyte Files			
d	1024	2048	4096	8192
0	0.969231	0.939406	0.882470	0.778706
1	0.030292	0.058727	0.110363	0.194867
2	0.000472	0.001828	0.006874	0.024287
3	0.000005	0.000038	0.000284	0.002010
4		0.000001	0.000009	0.000124
5				0.000006

Table 1: Fraction of columns with d differences.

w	$257-w$	$\binom{257}{w} \div 2^{257}$	w	$257-w$	$\binom{257}{w} \div 2^{257}$
129	128	0.049626	140	117	0.017813
130	127	0.048863	141	116	0.014718
131	126	0.047371	142	115	0.012074
132	125	0.045217	143	114	0.009710
133	124	0.042498	144	113	0.007687
134	123	0.039326	145	112	0.005991
135	122	0.035830	146	111	0.004596
136	121	0.032142	147	110	0.003470
137	120	0.028388	148	109	0.002579
138	119	0.024685	149	108	0.001888
139	118	0.021134	150	107	0.001358

Table 2: Probability of Hamming weight w or $257 - w$ for 257 Bernoulli trials, $p = 0.5$.

d	p	d	p
1	0.0498191	11	0.1381793
2	0.0500127	12	0.1386563
3	0.0751117	13	0.1503662
4	0.0753957	14	0.1508134
5	0.0943590	15	0.1618091
6	0.0947080	16	0.1623409
7	0.1106218	17	0.1726495
8	0.1110220	18	0.1732027
9	0.1250401	19	0.1829885
10	0.1254821	20	0.1835597

Table 3: Probability p of a d-bit difference changing the MAJORITY of 257 balanced i.i.d. bits.

d	Original Hamming Distance			
	1024	2048	4096	8192
1	0.19316	0.37449	0.70378	1.24264
2	0.00302	0.01170	0.04401	0.15547
3	0.00005	0.00037	0.00273	0.01933
4		0.00001	0.00009	0.00120
5				0.00007
S-Total	0.19623	0.38657	0.75059	1.41870
T-Total	50.236	98.962	192.151	363.188

Table 4: Expected number of differences d in the S- and T-arrays (256 x S).

d	Original Hamming Distance/Expected Hamming Distance between Alice and Bob's T-Arrays			
	1024/50	2048/99	4096/192	8192/363
0	0.717514	0.561947	0.398119	0.259184
1	0.203839	0.247257	0.240374	0.192400
2	0.057075	0.108175	0.144831	0.142723
3	0.015745	0.047054	0.087082	0.105797
4	0.004277	0.020348	0.052249	0.078368
5	0.001144	0.008747	0.031283	0.005801
6	0.000301	0.003737	0.018690	0.042907
7	0.000078	0.001587	0.011142	0.031714
8	0.000020	0.000670	0.006629	0.023424
9	0.000005	0.000281	0.003934	0.017288
10	0.000001	0.000117	0.002330	0.012750
11		0.000048	0.001377	0.009396
12		0.000020	0.000812	0.006919
13			0.000477	0.005091
14			0.000280	0.003744
15			0.000164	0.002751
16			0.000096	0.002019
17				0.001482
18				0.001086
19				0.000795
20				0.000582

Table 5: Probability of d differences in a column between Alice and Bob's T-arrays.

Continuing as before, we multiply each element in Table 5 by the probability that the given number of bit differences in a column will change the MAJORITY (Table 3) and multiply each entry by the number of columns, 128. The sum in Table 6 is the number we are seeking: the expected number of bits in which Alice and Bob's AMACs will differ, given the number of bits in the message that differ between them.

d	Original Hamming Distance/Expected T-Array Distance			
	1024/50	2048/99	4096/192	8192/363
1	1.299851	1.576717	1.532826	1.226906
2	0.365365	0.692480	0.927134	0.913641
3	0.151375	0.452387	0.837231	1.017161
4	0.041281	0.196367	0.504238	0.756302
5	0.013817	0.105642	0.377833	0.700623
6	0.003649	0.045305	0.226569	0.520146
7	0.001103	0.022472	0.157766	0.449056
8	0.000282	0.009518	0.094187	0.332868
9	0.000079	0.004495	0.062964	0.276691
10	0.000020	0.001880	0.037421	0.204781
11	0.000005	0.000857	0.024350	0.166184
12		0.000354	0.014406	0.122801
13			0.009190	0.097994
14			0.005409	0.072269
15			0.003398	0.056970
16			0.001992	0.041964
17				0.032740
18				0.024077
19				0.018632
20				0.013680
Total	1.876828	3.108473	4.816915	7.074447

Table 6: Expected Hamming Weight of $A \oplus A^*$.

Because the likelihood of seeing a Hamming distance of 10 or smaller by chance is less than 10^{-24}, one may say with some confidence that AMACs that differ in at most 5 bits are likely to have resulted from data that are 99.9% identical, and that AMACs that differ in at most 10 bits are likely to have resulted from data that are 99% identical. More precise statements of this sort would be desirable, so more work will be directed toward computing or estimating the variances as well as expected values for these distributions.

IMPLEMENTATION AND SIMULATION

The algorithm described above was implemented in C and tested on a Sun SPARCStation. We used a one megabyte Portable Graymap image file to test the effect of changing M on the T-array and AMAC. A was generated, a fixed number of bits in the image M were changed, then A^* was regenerated with the same K and I, and the two were compared. This was repeated 1000 times each for 1024, 2048, 4096, and 8192 bit changes. The results are shown in Table 7. Note that just slightly fewer bit changes in T^* were observed than predicted in Section 3. However, somewhat more changes in A^* were observed than predicted. (When the calculation was repeated with the rotations in step 7a, the results were similar.) Determining why we saw the larger differences in A^* will require further investigation.

	Hamming Distance between M and M^*			
	1024	2048	4096	8192
$T \oplus T^*$	48.56	96.09	185.19	351.23
$A \oplus A^*$	2.829	4.647	6.998	9.708

Table 7: Mean Observed Hamming Weights of $T \oplus T^*$ and $A \oplus A^*$ (1000 Trials).

No attempt has been made to optimize the implementation. More efficient data structures may be found, the calculation of the S-arrays can be done in parallel, and much of the cryptography can be pre-computed.

ADJUSTING PARAMETERS

If A is not sensitive enough to changes in M, two solutions are (1) to increment I, repeat, and construct the AMAC from multiple values of A, or (2) to replace each bit of A with bits containing more information about the Hamming weight of the k^{th} column of T.

For a general system, it is inconvenient to assume that the input has a fixed size or can be padded to a full megabyte. If the input is shorter than 1 megabyte, it could be padded with zeros; if it is longer, multiple 128 bit A values could be computed. Perhaps a better solution is to divide the length of the input by the length of A, round this up to the next odd perfect square to get the number of rows of M (the square root becomes the number of rows of S and T), pad M with zeros, and proceed as before. Clearly practical lower and upper bounds on the lengths of the input and of A exist.

DIRECTIONS AND OPEN QUESTIONS

Several extensions and open problems are:

- Alternative constructions, e.g., using other Boolean functions instead of MAJORITY.
- Performance measurements and improvements.
- Variations that pinpoint which locations in the input contain changes.
- Appropriate definitions of security for AMACs and provable security results.
- A similar technique in the public key instead of shared key model.

ACKNOWLEDGMENTS

The authors thank Gonzalo Arce for motivating discussions, Charles Boncelet for helpful comments on the statistical analysis, and Lisa Marvel for providing test images and a pleasant work environment at ARL.

REFERENCES

ARV99 Aiello, W., S. Rajagopalan, R. Venkatesan, "Design of Practical and Provably Good Random Number Generators," *Journal of Algorithms*, to appear.

AM97 Anderson, R., and C. Manifavas, "Chameleon —A New Kind of Stream Cipher," *Fourth International Workshop on Fast Software Encryption*, LNCS 1267, Jan. 1997, pp. 107–113.

Arc97 Arce, G., Personal communication, Nov. 1997.

ABGM99 Arce, G., C. Boncelet, R. Graveman, L. Marvel, "Applications of Information Hiding," *these proceedings*.

Bel96 Bellovin, S., "Problem Areas for the IP Security Protocols," *Sixth USENIX Security Symposium Proceedings*, USENIX Association, 1996, pp. 205–214.

DvOW92 Diffie, W., P. van Oorschot, and M. Wiener, "Authentication and Authenticated Key Exchanges," *Designs, Codes and Cryptography*, Vol. 2, 1992, pp. 107–125.

Fel68 Feller, W., *An Introduction to Probability Theory and Its Applications,* Volume I, Third Edition, John Wiley & Sons, 1968.

FIPS85 "Computer Data Authentication," FIPS PUB 113, NBS/NTIS, 1985.

FIPS93 "Data Encryption Standard," FIPS PUB 46-2, NIST/NTIS, 1993.

FIPS80 "DES Modes of Operation," FIPS PUB 81, NBS/NTIS, 1980.

FIPS95 "Secure Hash Standard," FIPS PUB 180-1, NIST/NTIS, 1995.

Lai92 Lai, X., *On the Design and Security of Block Ciphers*, Hartung-Gorre Verlag, 1992.

Pie93 Pieprzyk, J., and B. Sadeghiyan, *Design of Hashing Algorithms*, LNCS Vol. 756, 1993.

PGV93 Preneel, B., R. Govaerts, and J. Vandewalle, "Information Authentication: Hash Functions and Digital Signatures," *Computer Security and Industrial Cryptography*, LNCS Vol. 741, 1993, pp. 87–131.

Riv95 Rivest, R., "The RC5 Encryption Algorithm," *Second International Workshop on Fast Software Encryption*, LNCS Vol. 1008, 1995, pp. 86–96.

Sch96 Schneier, B., *Applied Cryptography, Second Edition,* John Wiley & Sons, 1996.

Sim91 Simmons, G., ed., *Contemporary Cryptology*, IEEE Press, 1991, pp. 381–419.