Algebraic Message Authentication Codes
Abhinav Aggarwal
Electronics and Computer Engineering Department
Indian Institute of Technology Roorkee
abhinav6891@gmail.com

Abstract
This paper suggests a message authentication scheme, which can be efficiently used for secure digital signature creation. The algorithm used here is an adjusted union of the concepts which underlie projective geometry and group structure on circles. The authentication is done through a key, which iterates over the complete message string to produce the signature. The iteration is not only based on the frequency distribution of the message string alphabet, but also on the probability of occurrence of another given reference string in the message. The complete process can be easily computed in a small time, producing signatures which are highly dependent on the message string. Consequently, the odds in favor of existence of a forgery are highly reduced.

Keywords
Algorithm, Message, Group, Projection, Operator, Key, Reference, Identifier

Introduction
Message authentication codes, commonly known as MACs, are efficient digital signature schemes, which can be used to authorize as well as authenticate a given piece of information, against any forger who attempts to alter its contents before its receipt at the receivers end. These codes verify the sender's identity, which is intelligently embedded in the MAC algorithm. The problem lies not in the space complexity, but in the time complexity. An efficient algorithm needs to be developed which reduces the time complexity, so that secure authentication tags can be generated in a feasible amount of time. The scheme suggested in this paper takes care of this problem and also ensures a secure embedding of the sender's identity (in the form of an identifier string provided by the sender) into the tag generated. So, the receiver, even on receiving the correct information, can verify if the transmitting medium is authenticated or not.

Motivation
To explain the need for AMACs, consider the following situation. Alice wants to send a message M to Bob, via a communication channel prone to attacks. This message needs to provide an authentication of Alice' identity to Bob, even if the message is received without any third party intervention. This means that Bob requires only Alice to send him this piece of information and not anyone else (Such a situation may arise during key exchange between two communicating parties, where the identity of one is of high importance to the other). Oscar, on the other hand, is a forger, who
attempts at attacking the channel and disrupts the originality of the message M. He can do it in two possible ways. He can either alter the contents of the original message, or somehow, alter Alice’ identifiers which were embedded in the MAC corresponding to the message. In either case, when Bob authenticates the information, he finds that a forgery has been attempted. We assume, for the moment, that the MAC scheme provides a mechanism, which is such that the probability of a forgery attempt clearing an authentication without any detection is practically zero. Hence, all attempts of Oscar fail to convince Bob against Alice. This paper is intended to help develop such a scheme. Apart from this, the use of stereographic projections and group structure on circles has been made to introduce a scheme in which the I/O alphabet and the processing data alphabet are different, so that the establishment of correspondence between the two also becomes a concern for the forgers. The two concepts, mentioned above, belong to two very different branches of Mathematics, namely Topology and Abstract Algebra, respectively, but their union can prove to be highly effective in producing secure MACs. Since the topics form the basis of modern Algebra, the scheme has been named Algebraic Message Authentication Code. A term frequently used here is Identifier for the sender. This refers to some piece of information, in the form of a string of message alphabet, which Alice would prefer embedding in the tag produced, so that when Bob tries to authenticate his message, he would also be sure of the sender’s identity to it. As an example, Alice may use the consonants in his name as the identifier ALC.

This identifier is shared between him and Bob, while Oscar is completely unaware of its existence. So, in essence, Alice and Bob have to now share a 2-tuple as their shared key, consisting of this identifier and a key. This way, Oscar’s forgery attempt on either or both of them does not hamper the authenticity of the message.

Mathematical Foundations
To implement an AMAC scheme, some primitive concepts of group theory and stereographic projections are required. The exact details are beyond the scope of this paper, but their usage is explained in this section. We will assume that the message string and the sender’s identifier are made up of the English alphabet, transmitted in the form of ASCII codes.

Consider a set S along with a binary operation * defined on its elements. If this binary operation (say addition, multiplication etc.) is closed, associative and assures the existence of unique identity and inverse elements for each member of the set, then the set is said to be a Group under that binary operation. It is denoted by the symbol <S,*>. If * is commutative too, the group is called an Abelian group. This much group theory is all what is required in the implementation. To understand basic stereographic projection in 1 dimension, consider an affine line L (say the real axis)
which is tangent to a circle $C$ at the point $P$, which coincides with the origin on $L$, as shown.

The domain of this projection is $L$, while the range is $C$. This means that points on the line are mapped onto the points on $C$. For any point $x$ on $L$, the image $f(x)$ is obtained by joining $x$ with a reference point $O$ on $C$, and obtaining the point of intersection with $C$. This point is the image of $x$ under the stereographic projection. The following points can be observed:

1. The point $P$ is invariant under this projection.
2. For every point on the line, there is a corresponding projection on the circle.
3. Points to the left of Point $P$ on the line $L$ are projected on the left-semicircle, while those on the right, project on the right semicircle.
4. The projection is continuous for all points on $C$.
5. Both the ends of the line (assume to be positive and negative infinity, in case of real axis) map to the point $O$. Hence, we assume that the line possesses these infinities as a single point and not two different points, thus maintaining the continuity of the projection at $O$.

6. For every point $y$ on $C$, there exists a unique $x$ on $L$, for which the projection of $x$ is $y$.

With these properties, let us define a binary operation $*$ on any points on the circle as follows:

1. Consider $A$ and $B$ to be two points on the circle, $C$.
2. Let $O$ be a special point, called the reference point, on $C$.
3. To multiply $A$ with $B$ (i.e. to compute $A*B$), join $A$ to $B$ with a straight line $AB$.
4. Draw a line parallel to $AB$, passing through $O$, and meeting $C$ at $Q$.
5. This point $Q$, is the value of $A*B$ with respect to $O$.

The above-defined binary operation $*$ forms an Abelian group with the set of all points on $C$. This operation is also commonly known as Point multiplication on circle, performed with respect to the point $O$, which acts as the identity for the group too. Using the above two concepts, we are now ready to define the algorithm underlining the AMAC scheme.

**Algorithm**

The algorithm for the AMAC scheme has been designed keeping in mind that the time complexity be low, the identifier embedment be secure and
as less prone to attacks as possible. The basic steps of the algorithm are as follows:

1. Let $S$ be the identifier string. The two communicating parties decide on this string as a one-time-communication-stamp. We will look for occurrences of the characters of this string (in order) in the message, for which we can either create a state machine, or use an equivalent logic. We assume that the function, $isRefChar(c)$, will return 'true' if the character 'c' passed is found in the reference string, in the required order.

2. Project the secondary key onto the circle to obtain the projection $K$. This will be embedded in the tag whenever $isRefChar()$ returns true.

3. All those characters in the message, which are not present in the reference string (either in face value or in the order specified) will be iterated as a Block, and the algorithm runs on such characters according to some Block Heuristic Function (BHF). This paper discusses two such functions; the design of better ones is left for future work.

4. The iteration of the algorithm is done from the first character of the message string, upto the very last. The domain of all computations is $R_{2\pi}$.

The complete algorithm is:

**AMAC ENCODE** :

**INPUT** : $msg\_string$ : string  
**OUTPUT** : $amac\_tag$ : double  

**EXTERNAL** :  
- $key\_K$ : double  
- $bhf(double[])$ : double  
- $isRefChar(char)$ : bool  
- $project(double,double)$ : double  
- $projectBack(double,double)$ : double  
- $multiply(double,double,double)$ : double

**STATIC** :  
- $cref$ : double  
- $block[0...n]$ : double

**CONSTANT** :  
- $PI = 3.14159$ : double

$amac\_tag = 0$  
$block[0] = 0$  
$cref = PI$  

$amac\_tag = multiply(amac\_tag, key\_K, cref)$  
$cref = PI - amac\_tag$

For each character $C_m$ in $msg\_string$

```plaintext
if (!isRefChar(C_m))
    append the ASCII code of $C_m$ to $block[1...n]$
else
    $block[0] = bhf(block)$
    $block[0] = project(block[0], cref)$
    $amac\_tag = multiply(ckey, block[0], cref)$
    $cref = PI - amac\_tag$

$amac\_tag = multiply(amac\_tag, ckey, cref)$

$amac\_tag = projectBack(amac\_tag, PI)$

return $amac\_tag$
```

The various functions and variables used are described as follows:

1. $msg\_string$ : The message to be authenticated; stored in the form of a string of characters
2. \( C_m \): Individual characters of the message string

3. \( \text{amac\_tag} \): To store the intermediate and final value of the authentication tag

4. \( \text{key\_K} \): Primary key projected onto the circle

5. \( \text{bhf()} \): Function to calculate a value corresponding to the contents of the array passed, based on some heuristic function

6. \( \text{project()} \): To get the projection of the point on the line, passed as a double. The projection is with respect to the point, passed as the second parameter.

7. \( \text{projectBack()} \): To get the preimage of the point on the circle, passed as the first parameter. The projection is with respect to the point, passed as the second parameter.

8. \( \text{multiply()} \): To multiply the points passed as the first two parameters, with respect to the point passed as the third parameter.

---

**Experimental Observations**

**Heuristic 1**

```cpp
double block_temp = 0;
double point;

for all i=0 to block length
    point = project( block[i], cref)
    block_temp = multiply( block_temp, point, cref)
return block_temp
```

This heuristic simply multiplies all the character-projections to the intermediate tag, and returns the final value obtained. This heuristic function produces codes which are highly dependent on the reference string, \( S \), and will not produce effective tags if \( S \) is not carefully chosen. For an arbitrary chosen \( S \), the probability that two intelligently chosen messages will produce the same tag is very close to 1, given that the forger has access to \( S \). In all other cases, this probability is very low, and is dependent on the frequency distribution of various alphabets and characters in an arbitrary-length English text.

The results for one simulation of this algorithm, over the above mentioned BHF, are shown below. The primary key used is “This is the first key.” (converted to the equivalent ASCII to obtain the equivalent in numeric format), and the secondary key used is “theveninester”.

Hope works in these ways: it looks for the good in people instead of harping on the worst; it discovers what can be done instead of grumbling about what cannot; it regards problems, large or small, as opportunities; it pushes ahead when it would be easy to quit;
Hope works in these ways: it looks for the good in people instead of harping on the worst; it discovers what can be done instead of grumbling about what cannot; it regards problems, large or small, as opportunities; it pushes ahead when it would be easier to quit; it "lights the candle" instead of "cursing the darkness." - Anonymous

As we can see, the three variations of the same message produce three different AMACs. It seems quite efficient in terms of sensitivity to even a single character change in the message. The running complexity of this heuristic based MAC is $O(n)$, where $n$ is the number of characters in the message string. But, this heuristic poses a potential problem. It is not prone to any forgery where the forger knows the secondary key, i.e. the reference string embedded in the tag. If he knows this key, he also knows the fact that all characters in the message string that lie between two consecutive letter in the reference string, can be permuted in any order to produce the same tag. This is due to the commutativity and associativity of point multiplication on a circle. The group is an Abelian group, and hence, all permutations fail to produce distinct codes. A careful selection of the secondary key, based on the frequency distribution of various characters in a general English text can be exploited. More frequently occurring sequences, if used intelligently, can reduce the impact of such a forgery. Thus, secondary key management becomes a key concern with such a heuristic.

Let us see another heuristic function and analyze it.

**Heuristic 2**

```c
bhf (block[]) : double
double temp = 0;
int B_HEUR = 10;
for all i=0 to block length
    temp = temp*(B_HEUR++) + block[i]
return temp
```

With this heuristic function, we have an integral incremental-multiplier to convert the sequence of characters in the block array into one single large number, which is returned by the function. This
method eradicates the chance of any two permutations producing the same tag at the end. The knowledge of secondary key only will be of no use to the forger. In practical use, this multiplier can be changed for every new message string, similar to a One Time Pad. Simulation results with this function are given below.

The primary key used is “Boomerang” (converted to a numeric equivalent using the ASCII codes) and the secondary key is “theveninthistle”.

Here's to the crazy ones. The misfits. The rebels. The troublemakers. The round pegs in the square holes. The ones who see things differently. They're not fond of rules. And they have no respect for the status quo. You can quote them, disagree with them, glorify or vilify them. About the only thing you can't do is ignore them. Because they change things. They push the human race forward. And while some may see them as the crazy ones, we see genius. Because the people who are crazy enough to think they can change the world, are the ones who do. Apple Inc.

AMAC -> 2.134199163829734402497706469148 3974456787109375

Here's to the crazy ones. The misfitings. The rebels. The troublemakers. The round pegs in the square holes. The ones who see things differently. They're not fond of rules. And they have no respect for the status quo. You can quote them, disagree with them, glorify or vilify them. About the only thing you can't do is ignore them. Because they change things. They push the human race forward. And while some may see them as the crazy ones, we see genius. Because the people who are crazy enough to think they can change the world, are the ones who do. Apple Inc.

AMAC -> 0.3831324289500477875414219 39749852754175662994384765625

Again, it can be seen that the codes are highly sensitive to even single character changes. But unlike the previous heuristic function, this code is highly dependent on the order of characters in the message string. The probability that two sequences produce the same tag,
even if the two are intelligently designed, is very low.

**Conclusion and Future Work**

The algorithm discussed above can be greatly modified by the use of intelligent designed keys and heuristics for handling message blocks. A basic version shown in this paper highlights the essence of how this scheme will be implemented. Use of Galois theory and Field theory will eradicate the use of any kind of lookup tables, required for standard hashing techniques. The running complexity is thus, greatly reduced. All work, thus, needs to be concentrated on authentication mechanism and key exchange.

**References**

1. S.P. Eugene Xavier, *Theory of Automata, Formal Languages and Computation*, New Age International Publishers, 2008
2. Joseph Rotman, *Galois Theory*, Second Edition, Springer, 2010
3. John B. Fraleigh, *A First Course in Abstract Algebra*, Pearson, 2011
4. Douglas R. Stinson, *Cryptography Theory and Practice*, Third Edition, Taylor and Francis Group, LLC, 2006