Synopsis

for

Masters of Technology Project

" YOGA KARMASU KAUSALAM "

Under the guidance of

Prof. D. Roychowdhury

Department of Computer Science and Engineering,

Indian Institute of Technology Kharagpur

By:

Umang Jain (03CS3005)

1. Framework for online Authenticated Encryption using Cellular Automata

1.1 Introduction

The conventional view had been that encryption with a block cipher gave "a bit of protection" to the integrity of messages, and that encryption of data was sufficient to provide privacy in all cases. These perceptions were false as we all know now. Therefore the need for authentication along with encryption cannot be emphasized more. Certain applications need online data. If they employ conventional Authenticated Encryption schemes under standard setup they have an inherent drawback as they would have to wait for complete message to arrive to verify the integrity of the message. If the data has to be used in an online fashion it will most likely be data whose integrity has not been established. Therefore there is the need for an Authenticated Encryption that is online on true sense.

1.2 Objective

In the present work we look at the problem of encryption accompanied with authentication in an online manner. There is a need for a framework for the above mentioned task which we propose in the current work. We have also proposed a hash function based on cellular automata compatible with the above mentioned framework.

1.3 A framework for online Authentication and Encryption

The framework for message for the authenticated encryption is outlined below.

- Data is divided into blocks of 128 bits each.
- Encryption is block is done by using a block cipher with block size 128 bits.
- Blocks of messages (128 bit each) are encrypted and the cipher-text is sent to the receiver
- An authentication tag or hash value is computed for a group of 8 blocks
- The hash value is also 128 bit in size

- After 8 message blocks are sent , the hash value is encrypted and sent
- The receiver on receiving message blocks performs similar operations to compute the hash value
- The receiver on gets the encrypted tags after 8 consecutive encrypted message blocks
- Computed and decrypted hash values are compared to verify the integrity of data

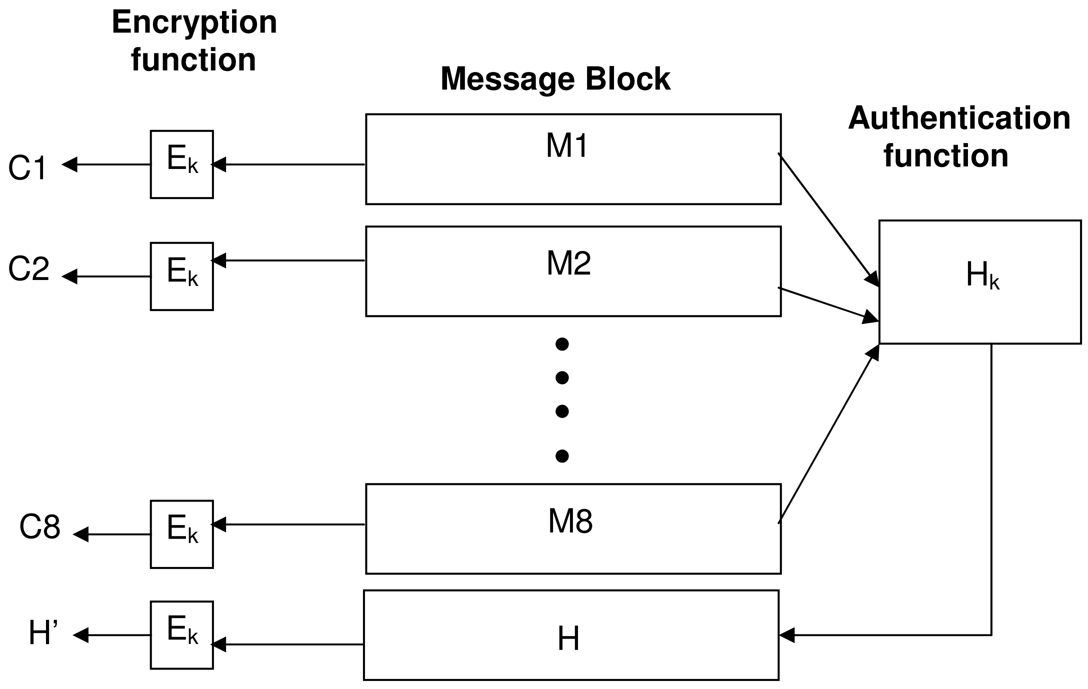

Fig 2.1: AE scheme(block diagram)

1.4 CHASH: A cellular automata based hash function

The complex behavior and the complex patterns that CA's generate in spite of their fairly simple structure and update rules make them an interesting prospect for application in the field of cryptography. With, CHASH we try to tap these interesting properties of Cellular Automata in a hash function.

1.4.1 Overview

CHASH is keyed hash function. The input to the function is the plaintext. The plaintext is padded and divided into blocks of 1024 bits. A hash value is computed for each of the 1024 blocks. For this each of the 1024 bit block is further divided into sub-blocks and these sub-blocks are operated on by some CA rules. The basic algorithm can be outlined as follows:

- Rule generation using the key
- Message is padded such that message is 64 modulo 128. For this a 1 followed by required number of 0's is added. Then length of the message encoded in 64 bits is added to make the length a multiple of 128 bits.
- Divide the message into 1024 bit blocks
- Divide each 1024 bit block into eight 128 bit sub-blocks.
- Run CA rules on the first 128 bit block to get intermediate hash.
- XOR the intermediate hash and next 128 bit block
- Continue the last two steps until the last block
- The eighth intermediate hash is the final hash for this 1024 bit block.
- For 128 blocks that are left over after dividing message into 1024 bit blocks, hash is computed separately.

1.4.2 Rule Generation

The rule generated by using the key is not any standard CA rule. It is rule with radius =3 or neighborhood =7. It means that each bit in the next state directly depends on 7 bits of previous state.

We will first have a look at CA rule 30. It can be described as:-

Now for a 7 neighborhood CA rule the rule-table must consist of $2^7 = 128$ entries. Each entry will either hold a 0 or 1. The rule table is generated using following steps:

- We generate a 256 entity by using the key. Let us call this the intermediate rule table. First 128 bits are same as 128 bits of the key. The remaining 128 bits are generated by negating the bits of the key.

  Intermediate rule table[i] = key[i], 0<= i <128

  = key[i] XOR 1, 128<= i <256

- This rule-table is run on CA rule 30 and on right-toggle version of CA rule 30 alternatively for n1 rounds (where n1=64) rounds. We then pick 128 bits from this 256 bit state to get the final rule-table.

The basic idea behind the steps remains the same as in original CHASH. Rule 30 is applied because it is non linear, has a high complexity of inversion for arbitrary seed values and has very good randomness properties The method of generation of the intermediate rule-table ensures that the rule has nearly equal number of 0's and 1's. Both rule 30 and the modified rule 30 are applied for good diffusion and remove any patterns that might lead to discovery of the key.

1.4.3 One round of CHASH

One round of CHASH can be described as follows:

1. Perform modulo 2 addition of last round's output and the message block. If the block is the first message block add modulo 2 the initialization vector.
2. Transform the result of step 1 using the rule table for n2 cycles (where n2=20).
3. Transform the result of step 2 using the CA rule 30.
4. Transform the result of step 3 using the right toggle CA rule 30.
5. Repeat step 3 and 4 for n3 cycles(where n3 = 10).
6. Output the result of step 5.

1.4.4 Generation of the hash value

Each of the 128 bit sub-blocks of a 1024 bit block is run one round in succession with chaining and the final hash output is the hash value for this 1024 bit block.

1.4.5 Design Rationale

- Based on Cellular Automata**:** The use of cellular automata makes the algorithm easier for implementation in hardware it's regular, modular and cascadable structure.
- Padding: Since in the padding scheme, we also append the length of the message, it foils attacks like message extension and fixed point attacks.
- Output Length**:** The output hash length currently is 128 bits which is secure for a keyed-hash (key length = 128). For future also both hash length can be simultaneously increases. For example if we want increase the length of the hash to 256 bits, we make the key-length 256 bits as well. For this all that we need to do is to make the rule as 8 neighborhood rule. Apart from this nothing needs to changes in the algorithm.
- Non-Linearity: The use of CA rule 30(which is a non-linear rule) introduces non-linearity in the algorithm. Due to this the differential corrective patterns are not preserved which makes differential attacks against CHASH very difficult.
- Balanced-ness: The rule that is generated using the key has almost the same number of 0's and 1's.
- Both CA rule 30 and right toggle CA 30 are applied to annul the effect of propagation of difference only to the left, when rule 30 is applied.
- The unknown transformation based on the rule table is applied first before applying rule 30 so that even with a chosen plaintext attack it is not possible to guess the internal state of the CHASH round. This means that the collision attack of the type mentioned in 4.2.2 are no longer possible.
- Initialization vector is used to prevent any weakness due to weak messages like the all 0 message. Now it will not be possible to predict the output of an all 0 message.
- Total number of possible rules of 7 neighborhood is 128, so it is not possible to guess the rule that is generated using the key.
- Both the initial transformation and the last applied to a message is unknown to the attacker, this also lends strength to the hash function.

1.5 Conclusion

CHASH, the hash function proposed in this work is a flexible easy to implement algorithm. In sync with the pre-proposed framework the output length and key length can be changed according the requirement of the application. Further in this work we have carried out detailed security analysis and statistical evaluation of CHASH. Results show that CHASH performs satisfactorily on the statistical tests.

2. Anonymous Authentication in VANETs

(This work has been done jointly with Nitin Bansal (03CS30015))

2.1 Introduction

Vehicular networks are likely to become the most relevant form of mobile ad hoc networks. Thus address security of these networks become paramount. Manufacturers are about to make a quantum step in terms of vehicular IT, by letting vehicles communicate with each other and with roadside infrastructure; in this way, vehicles will dramatically increase their awareness of their environment, thereby increasing safety and optimizing traffic. There are many aspects to vehicular communication in terms of implementation and inherent challenges. One of these challenges is security; very little has been devoted so far to the security of vehicular networks. Yet, security is crucial. For example, it is essential to make sure that life-critical information cannot be inserted or modified by an attacker; likewise, the system should be able to help establish the liability of drivers; but at the same time, it should protect as far as possible the privacy of the drivers and passengers.

2.2 Objective

In this work we have addressed the problem of entity authentication in VANET's. For every received message the recipient should be able to verify whether the information sent has been sent by an authentic sender or not. On the other hand it is also paramount that the identity of the sender is not revealed through his signature (the Anonymity requirement). Though if the need arises some trusted party or a law enforcement agency should be able to establish liability through the signature.

2.2.1 Infrastructure and Network Assumptions

The communicating nodes in VANETs are either vehicles or base stations (Road Side Units). The communications are assumed to be like those in DSRC (Dedicated Short Range Communication). All the vehicles broadcast. They can send to and receive messages from vehicles within 1000 meters distance. Messages are sent every 100ms. Messages can be event driven or periodic information about a vehicles position, speed etc.

2.3 State of the Art

More attention is being paid to the security of VANET's these days. Having said that, there is still no authentication algorithm that has been developed keeping VANET and its needs in minds. There have been many suggestions about security protocols for VANET's all of which seem to use already established and secure algorithms like RSA and ones based on these. But the major bottleneck in VANET is that vehicles are not expected to have huge computational or storage resources. Therefore schemes RSA or those based on Elliptic Curve Cryptography do not make for very suitable candidates for authentication in VANET. In this work we explore the possibility of using a group signature scheme for authentication in VANET. In the next section we propose a novel group signature schemes based on the Chinese Remainder Theorem.

2.4 Group signature scheme based on the Chinese Remainder Theorem

Let there are k group members.

Public Information (known to all members and manager)

N_G - A relatively prime number

Group Manager has following information:

N_0 - Private relatively prime number (known only to manager) used to reveal identity of the message sender

Group Members:

Each member M_i is given the following information by the group manager.

N_i - A relatively prime number known only to M_i

a_i - A random number ($< N_i$) known only to M_i used for sign verification

$Pr_i = \prod (N_j)$ where j!=i

N_G - A prime number known to all members.

Each N_i is relatively prime to each other.

$CRTK_i$ which is created as follows:

$CRTK_i \bmod N_0 \equiv ID_i$

$CRTK_i \bmod N_1 \equiv a_1$

$CRTK_i \bmod N_2 \equiv a_2$

....

$CRTK_i \bmod N_k \equiv a_k$

$CRTK_i = < ID_i, a_1, a_2, a_3, a_4 \ldots a_k >$ (k Tuple) as in CRT

The modulus is taken with respect to N_0 and all other N_j.

All this information is available with a particular member.

Note: *$CRTK_i \bmod N_i \equiv a_i$ is not used in creation of $CRTK_i$*

2.4.1 Signature Generation

To send the message the member creates a signature Y in the following manner.

$Y \bmod Pr_i \equiv CRTK_i$

$Y \bmod N_G \equiv$ Hash (Message)

$Y = < CRTK_i,$ Hash (Message) $>$

2.4.2 Signature Verification

To verify the signature a member M_j does the following -:

$X = Y \bmod N_j$

If ($X == a_j$) the signature is verified.

It is important to note that the verifier does not need to and cannot extract $CRTK_i$ of the sender, to verify the authenticity of the sender.

2.4.3 Identity Extraction

Only Manager will be able to reveal the identity of message sender by doing following operation:

$ID_i = Y \bmod N_o$

This ID_i then can be mapped to the actual identity of the sender.

Note: *$N_0, N_1, N_2, \ldots\ldots N_k, N_G$ they are all relatively prime to each other.*

2.4.4 Correctness

In order to verify the receiver does the following check -:

If($Y \bmod N_i == a_i$)

We have to prove that in case of an authorized sender, this check does stand to be true.

$CRTK_i = (\sum a_j * ((Pr_i/N_j)* (((Pr_i/N_j)^{-1} \bmod N_j)))) \bmod Pr_i$ -------------- (1)

Where j varies from 0 to k and j!=i

$Y = (CRTK_i (N_G * (N_G^{-1} \bmod Pr_i))+Hash<Message>(Pr_i*(Pr_i^{-1} \bmod N_G))) \bmod Pr_i*N_G$ -(2)

Let Z be a number such that

$Z \bmod N_0 \equiv ID_i$

$Z \bmod N_1 \equiv a_1$

$Z \bmod N_2 \equiv a_2$

....

....

$Z \bmod N_k \equiv a_k$

$Z \bmod N_G \equiv Hash<Message>$

Let $N_{k+1} = N_G$, $a_0 = ID_i$, $a_{k+1} =$ Hash<Message>

Let $P = Pr_i* N_G$

Therefore Z can be written as-:

$Z = (\sum a_j * ((P/N_j)* (((P/N_j)^{-1} \bmod N_j)))) \bmod P$ where j varies from 0 to k+1 and j!=i

$Z = (\sum a_j * ((P/N_j)* (((P/N_j)^{-1} \bmod N_j)))) \bmod P + a_{k+1} * (Pr_i * (Pr_i^{-1} \bmod N_G)) \bmod P$,

j varies from 0 to k and j!=i

Let Z = Z1 + Z2, where Z1 and Z2 are the two terms in the above equation

$Z1 = (\sum a_j * ((Pr_i*N_G/N_j)* (((Pr_i*N_G/N_j)^{-1} \bmod N_j)))) \bmod P$

Since Pr_i is a multiple of N_j,

$N_G^{-1} \bmod N_j = N_G^{-1} \bmod Pr_i$

$Z1 = (((\sum a_j * ((Pr_i/N_j)* (((Pr_i*/N_j)^{-1} \bmod N_j)))) *(N_G*(N_G^{-1} \bmod Pr_i)))) \bmod P$ – (3)

Now we have from equation 1

$\sum a_j * ((Pr_i/N_j)* (((Pr_i*/N_j)^{-1} \bmod N_j)) = qPr_i + CRTK_i$ for some integer q ----- (4)

From (4) and (5)

$Z1 = ((qPr_i + CRTK_i)* N_G*(N_G^{-1} \bmod Pr_i)) \bmod P$

$= (((qPr_i*N_G + CRTK_i * N_G) \bmod P * (N_G^{-1} \bmod Pr_i) \bmod P) \bmod P$

$= (((qP + CRTK_i * N_G) \bmod P * (N_G^{-1} \bmod Pr_i) \bmod P) \bmod P$

$= (((CRTK_i * N_G) \bmod P * (N_G^{-1} \bmod Pr_i) \bmod P) \bmod P$

$= ((CRTK_i * N_G* (N_G^{-1} \bmod Pr_i)) \bmod P$

$Z1 = ((CRTK_i * N_G* (N_G^{-1} \bmod Pr_i)) \bmod P$ --------- (5)

$Z = Z1 + a_{k+1} * (Pr_i * (Pr_i^{-1} \bmod N_G)) \bmod P$

$= ((CRTK_i * N_G* (N_G^{-1} \bmod Pr_i)) \bmod P + Hash<Message> * (Pr_i * (Pr_i^{-1} \bmod N_G)) \bmod P$

$Z=((CRTK_i * N_G* (N_G^{-1} \bmod Pr_i)+Hash<Message> * (Pr_i * (Pr_i^{-1} \bmod N_G))) \bmod Pr_i* N_G$

= Y from (2)

Therefore Y = Z

Hence,

$Y \bmod N_j = Z \bmod N_j = a_j$

2.5 Application to VANET

We assume the VANET to be divided into several groups with one trusted party certifying authority) acting as a Group Manager for each group. The Group Manager thus is not a vehicle but some sort of a government agency. Each vehicle or member of a group shall be given public and private information by the Group Manager at the onset. Each of them can sign and verify messages as mentioned above.

2.5.1 Communication Overhead

Let each N_i's have size b bits and there are k members. CRTK's and Pr's will have size of the order of b*k bits. This poses a problem for large groups as CRTK and Pr will lead unacceptable size requirements. The overhead will be the size of Y.

For b = 80 bits and k = 10000

Overhead is of the order b*k bits = 100*8 kilo bites = **100 Kilo bytes**

2.5.2 Storage Overhead

We need to store following things for this proposal

CRTK, Pr , N_i and corresponding a_i and N_G

Order of Storage overhead =100 kilo bytes (CRTK) + 100 kilo bytes (Pr) + 160 bits (N_i's and corresponding a_i's) + 80 bits (N_G) = **200 kilo bytes.**

2.6 Conclusion

Evidently the overhead and storage requirements of the scheme are huge at the moment, even to the scale of being impractical. Intuitively the message signing and verification seems to be faster than those in RSA which involve exponentiation. But the huge size of the signature generated might well nullify this intuition as well. Further in this work we show that we are right in saying that the scheme presently in impractical both time-complexity and storage complexity wise. After that we modify the scheme to drastically reduce the amount of storage and overhead and we show that while signature generation time is comparable to that in RSA, the verification time is less which is significant as in VANET vehicles are likely to verify more often that they sign. We also provide a detailed security analysis and comparison with other VANET protocols.