Lightweight Broadcast Authentication Protocols Reconsidered

Shigenori Yamakawa*, Yang Cui†, Kazukuni Kobara† and Hideki Imai+†

*Department of Electrical, Electronic & Communication Engineering
Chuo University, Japan
Email: shigenori-yamakawa@mailab.jp
†Research Center for Information Security (RCIS), National Institute of Advanced Industrial Science & Technology (AIST), Japan
Email: {y-cui, k-kobara, h-imai}@aist.go.jp

Abstract—In the emergency broadcast system (or emergency alert system) which aims to broadcast a warning information immediately in time of emergency such as a natural or civil disaster, computational power-restricted devices such as, pocket terminals and sensors need to instantly and securely verify correctness and integrity of the received message packets. Though a lot of broadcast authentication systems were proposed, most of them require relatively high computation cost.

In this paper, we propose a new lightweight broadcast authentication protocol McSBA based on McEliece signature. It can be quickly verified with a tiny computation cost, applicable on power-restricted devices. We first estimate the time performance and compare McSBA with widely used RSA signature and well-known TESLA broadcast authentication protocol, to show that the verification of McSBA has a low cost and is faster than the others. Especially, it is shown by our estimation that McSBA can verify quickly less than 1s in emergency situations, however RSA signature with the same security takes more than 4s. Then we make use of a simulation of verification of RSA signature and McSBA, on the same platform, to attest that verification of McSBA is about ten times faster than RSA signature, which also supports our estimation result. Consequently, we expect that our technique is useful in the emergency broadcast system.

I. INTRODUCTION

As we all know, the broadcast communication style is recently popular all over the world. Broadcast communication method is convenient in that one packet is efficiently transmitted to unspecified number of receivers. But adversaries who wants to block or falsify the information broadcasted, can attack more easily in one-to-many communications as compared with in one-to-one communications. Hence, it is crucial for each receiver to verify that whether the received data has been forged, or modified through broadcast communication. In order to solve this problem, it requires efficient broadcast authentication protocols corresponding to broadcast communication eventually.

In wireless environment, it is furthermore desirable to achieve the security with low cost devices, such as sensor networks, besides the efficient authenticity. As one of solutions, the TESLA protocol [1] was proposed by Perrig, Canetti, Tygar and Song in 2002. TESLA is widely known as a lightweight broadcast authentication protocol. Because easy-to-compute MAC (Message Authentication Code) and hash function is used as the authentication technique in the protocol, it enables TESLA to take low computation effort to broadcast.

By use of “time delay” that buffering the packets and a tight time synchronization, TESLA can be tolerant with packet loss, as well. TESLA is a useful authentication protocol, however, it may not play well for authenticity of emergency broadcast due to the heavy dependence on time delay.

For example, the broadcast in time of a natural or civil disaster must make pocket terminals verify authenticity of transmitted information instantly. However, the power of pocket terminals is too marginal to operate expensive verifications of the transmitted packets. In addition, one problem from buffering and time synchronization, and another problem from time delay to verify the information, are derived from the specific property of TESLA. Time delay to verify the information is a crucial issue and not desired when considering emergency. For example, consider the earthquake scenario, which frequently happens in some areas and countries. Assume the alerting broadcast comes as soon as the first wave of earthquake, then a quick response within one second will be of great importance since sensors can shut down the dangerous electricity and gas power on time. A couple of seconds delay may lead to fatal incidents in practice. In order to solve these delay problems, we would like to build quickly verifiable broadcast authentication with a low computation cost. Therefore we hereby reconsider the broadcast authentication protocols and put forward ours.

In this paper, we propose a new lightweight broadcast authentication based on McEliece digital signature [2]—McSBA, and estimate the computation time of McSBA as compared with TESLA and widely used RSA signature (RSA signature is faster than Elliptic Curve signature to verify [3], hence we pick up RSA signature as our competitors). As a result of the estimation, we show that the proposed McSBA protocol can verify much faster than TESLA and RSA signature in low power devices (such as ATmega128L 4MHz chip, according to the recent references), under the same security level of about $2^{90}$ complexity. Especially, it is shown by our estimation that McSBA can verify quickly less than 1s in emergency situations, however RSA signature with the same security takes more than 4s. And our simulation result based on the same
platform also supports the above estimation, that McSBA is several ten times faster than RSA signature with the same security level.

The result implies that McSBA is useful as an emergency broadcast authentication protocol in terms of the verification cost performance on the low power device.

We first provide the proposed lightweight broadcast authentication McSBA protocol in Sec.2. Then, we estimate the computation time of the three broadcast authentication protocols in Sec.3, and simulate the computation time of both McSBA and RSA signature in Sec.4.

II. THE PROPOSED PROTOCOL: LIGHTWEIGHT BROADCAST AUTHENTICATION MCSBA PROTOCOL FOR EMERGENCY COMMUNICATION

The original McEliece public-key cryptosystem (PKC) was proposed by McEliece [4] in 1978. McEliece PKC has a good performance of fast encryption and decryption. McEliece digital signature [2] enjoys its fast verification. Unlike RSA nor ElGamal, McEliece PKC is believed to be secure against corruption by verifying the signature. The reason of that no disclosure of trapdoor $\varphi$ to be close to NP-hard). McSBA is expected as one of promising signature schemes which guarantee the long-term security.

A sender attaches signature to every packet by use of her signing key. Our broadcast authentication protocol based on digital signature enables receivers to detect the falsification or corruption by verifying the signature. The reason of that no use of digital signature on low cost devices as time being, are heavy computation cost which involved. But by our use of McEliece signature [2], lightweight computation is achieved. Furthermore, we also modify to the McEliece signature so that it can be verified more quickly. Finally, another interesting result of using our proposal instead of TESLA is that (signature) broadcast authentication protocol can be made in advance. Note that $T$ can be made in advance.

Let $n$ denote a code length, $k$ denote a code dimension in a binary Goppa code correcting $t$ errors. Let $z$ denote a binary $n$-bit vector with Hamming weight $t$.

The McEliece Signature Broadcast Authentication protocol McSBA is described as follows.

**McSBA=(Setting, Broadcast, Authentication)**

1) **Setting:** A sender sets up the following offline.

   a) **Gen:** On input $\kappa$, output $(sk, vk)$, $n, t \in \mathbb{N}$, $t \ll n$
   
   - $sk$ (Signature Key): $(S, \varphi, P)$
   
   - $G'$: $k \times n$ generating matrix of a binary irreducible $[n, k]$ Goppa code which can correct a maximum of $t$ bits.
   
   - $H'$: $(n-k) \times n$ parity check matrix given by $G' H'^T = 0$.
   
   - $\varphi$ is an efficient bounded distance decoding algorithm of the underlying code (for syndrome decoding).
   
   - $S$: $(n-k) \times (n-k)$ non-singular matrix, chosen at random.

   b) **Sig:** On input message $M$, output signature $[\cdots I_z \cdots | i_o | T]$ and time stamp $T$.

   - input the anticipated message $M$ for emergency communication.
   
   - extract the hash value $s = h(M)$ by hashing $M$.
   
   - compute the hash value $s_i = h([\cdots s \cdots | i_i])$, by means of hashing $[\cdots s \cdots | i_i]$ for $i = 0, 1, 2, \ldots$
   
   - find the minimum value $i_o$ of $i$ such that the syndrome $s_i$ is decodable.
   
   - compute the error pattern $z$ of weight $t$ by use of trapdoor $\varphi$ solving syndrome decoding problem is thought to be close to NP-hard).
   
   - compute the index $I_z$ of $z$ in the space of words of weight $t$.

   - $I_z = (i_1 | i_2 | i_3 | \cdots | i_t)$

   - $(i_3 < \cdots < i_t)$ means non-zero positions of $z$.

   - attach the signature $[\cdots I_z \cdots | i_o | T]$ to the message $M$.

   - prepare time stamps $T$ for preventing from replay attack. Note that $T$ can be made in advance.

   - disclose a parity check matrix $H$ and a hash function $h$ to verify the transmitted packet.

2) **Broadcast:** A sender broadcasts the packet $P = \{M || [\cdots I_z \cdots | i_o | T]\}$. And a time stamp changes at a constant time in order to prevent from replay attack.

3) **Authentication:**

   - receive the packet $P$.

   - verify whether the attached time stamp $T$ is right or not. This could be done quite efficiently by hash-and-compare.

   - parse the input into the message $M$ with signature $[\cdots I_z \cdots | i_o]$.

   - recover the $n$-bit error pattern $z$ from its index $I_z$.

   - compute the syndrome $s_1 = z H'^T$ by use of public-key parity check matrix $H$.

   - compute the syndrome $s_2 = h([\cdots h(M) \cdots | i_o])$ by use of hash function $h$.

   - verify whether the signature is right or not to compare $s_1$ with $s_2$.

Finally, compare $s_1$ with $s_2$. When $s_1 = s_2$, we can see that its signature is valid.

Note that we have changed McEliece signature (which indeed makes it a little bit longer) to achieve fast verification. Rather than sending compressed syndrome, we are sending the exact positions of the non-zero bits (total $t$ positions) to speed up the verification process. The time stamp has validity in a

---

1We here make use of a different encoding method other than CFS’s [2] and Cover’s [5]. Details will be explained in the full version.
A. Chosen Parameters

McEliece signature scheme given in [2] selects new security parameters \( n = 2^{16} = 65536, t = 10 \) in a binary Goppa code structure. Though signing algorithm takes at most \( t! \) times of decoding to successfully generate one valid signature, it is efficient to compute when \( t = 10 \).

B. Security Parameters Setting

We would like to pursue a security level around \( 2^{90} \). For a given integer \( m \), we use a \((n = 2^m, k = n - tm, d \leq 2t + 1)\) binary Goppa code such a tuple of security parameters \((n = 2^{17} = 131072, k = 130885, t = 11)\). According to Table 2 in [2], this achieves a \( 2^{90} \) security level.

C. Applied Scheme

We propose the signature scheme that the length of signature \( \cdot \cdots I \) is \( 187 + \log_2 (t!) \approx 213 \) bits. (Since \( z \) is \( n \)-bit long, and there is exact \( t \) non-zero positions included in \( z \), \( I_2 \) could always be expressed as \( t \times \log_2 n = 11 \times 17 = 187 \) bits.) In this scheme, all error positions are transmitted. Hence searching error positions is not necessary for authentication and it enables this scheme to perform a fast verification. In the McEliece signature by Courtois et al [2], their motivation exists in shortening the signature length. Hence, they employed another encoding method for \( I_2 \) than us.

However, our proposed McSBA protocol is aimed at speeding up the verification. Hence we recommend the above encoding method whose operation is simple introduced in the authentication part of McSBA. Note that we can further reduce this length but increase the generation time of signature.

D. Properties

One good property of McEliece encryption is its fast and simple operations, including some binary multiplication and bit-wise exclusive-OR. Analogously, it has similar cost in verification of signature.

The other good property of McSBA is the short 213-bit signature length. It will play an important role in ensuring authenticity about emergency information in circumstances where we are permitted to use only low power devices.

E. Security Analysis

The most powerful attack on McSBA protocol is the FLWC (Finding Low-Weight Codeword) attack [6] and its binary work factor for \((m, t) = (17, 11)\) is estimated to be \( 2^{90.4} \) [7].

III. Estimation about Computation Time

Assume the broadcast authentication in emergency situations. Commonly, a sender can make use of a powerful server to manage a group of low power devices, and receivers are low power devices such as the pocket terminals. In the following, we estimate efficiency from a point of computation time view, of broadcast authentication protocols, McSBA and TESLA. We also include 2048-bit RSA digital signature in the comparison as a criterion.

We evaluate the signing time of those protocols on Pentium III 1GHz CPU. On the other hand, verification time on the low power device is the estimated value in the case of using ATmega128L 4MHz chip. Table 1 shows the estimation result.

According to [8], the computation time of SHA-2 in ATmega128L is 170ms. Consequently, to construct HMAC-SHA-2, we take that computation time as 425ms which is normally 2.5 times of original hash computation.

A. 2048-bit RSA signature

On the security issue, we consider not 1024-bit but 2048-bit RSA signature for the near future. For 2048-bit, security level is estimated to around \( 2^{90} \) complexity [9] which is considered secure in next ten years.

1) Applied Hash Function: We would use SHA-2 instead of SHA-1, because it is shown that SHA-1 seems to be insecure according to Wang et al. [10]

2) Signing Time: According to (2), Sec.2.3. in [11], signing time is 43.12ms by converting CPU power 800MHz to 1GHz.

3) Authentication Time: According to (12), it takes 3.89s for authentication in the implementation of ATmega128L 4MHz chip. This time is too long to authenticate normally in emergency broadcast. Besides that, one computation of hash function (SHA-2) is counted. According to [8], it takes 170ms to calculate SHA-2 on the ATmega128L chip.

B. New Protocol McSBA

The McSBA protocol adopts signature scheme in Sec.2.C. (assuming that security level is around \( 2^{90} \) complexity). Unlike TESLA, time delay is not required for correct authentication.

1) Applied Hash Function: For comparison, we also use SHA-2.
2) Signing Time: Since McSBA is based on asymmetric techniques, and only a lightweight computation is needed on the receivers’ side, a little complicated preparation is naturally required for the sender. According to [2], to generate a valid signature, takes from 10s to 30s to run with a Pentium III 1GHz CPU\(^2\) when \(t = 9\). For the case when \(t = 11\), it takes 143 times of computation to get the signature, because the rate of signature cost between \((m, t) = (17, 11)\) and the previous \((16, 9)\) is \(11 \times 10 \cdot (11/9)^2 \cdot (16/16)^3 \approx 143\) (signature cost is given as \(t!t^2m^3\) in [2]). This signature generation time does not cause any delay in verification since signatures can be generated in advance. Hence, we can say that generation of signature could be practically dealt with.

3) Authentication Time: According to [2], it takes less than \(1\mu s\) for authentication in Pentium III CPU 1GHz. However, this result [2] seems not to include finding the other syndrome \(s_2\).\(^3\) The reason comes from that it must count hash computation by two times in order to find \(s_2\).

Thus, we convert the result of [2] (with 32-bit Pentium III CPU 1GHz), into case of 8-bit ATmega128L 4MHz. Since all are binary exclusive-OR operation, we simply count by the power of CPU, such as expansion rate of \((1000/4) \cdot (32/8) \approx 1000\). It means that \(1\mu s\) computed by Pentium III CPU 1GHz corresponds to \(1\ms\) over ATmega128L. Even considering errors, we still have a big margin about several ten \(\text{ms}\).

As a result, we show that it takes \(340 + 1 = 341\ms\) for authentication in the low power devices equivalent to ATmega128L chip 4MHz. This authentication time is very fast as compared with popular 2048-bit RSA signature.

TESLA needs a frequent update of the keys as a hash chain. Unlike TESLA, RSA signature and McSBA only run the calculation on emergency, which saves the power consumption.

\(^2\)By using high bench-mark CPU, we can get better result.

\(^3\)If their result is true, then our scheme can reduce the verification time further.

<table>
<thead>
<tr>
<th>Protocol</th>
<th>2048-bit RSA signature</th>
<th>213-bit McSBA ((n = 131072, k = 130885, t = 11))</th>
<th>TESLA</th>
</tr>
</thead>
<tbody>
<tr>
<td>Operations</td>
<td>RSA Signature + SHA-2</td>
<td>Bit-wise XOR + 2(\times) SHA-2</td>
<td>2(\times) SHA-2 + HMAC-SHA-2</td>
</tr>
<tr>
<td>Length[bit]</td>
<td>2048</td>
<td>213</td>
<td>180(^*)</td>
</tr>
<tr>
<td>Frequent Update</td>
<td>Not necessary</td>
<td>Not necessary</td>
<td>Necessary</td>
</tr>
<tr>
<td>Tight Synchronization</td>
<td>Not necessary</td>
<td>Not necessary</td>
<td>Necessary</td>
</tr>
<tr>
<td>Verification Delay</td>
<td>No</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td>Hash-chain</td>
<td>Not necessary</td>
<td>Not necessary</td>
<td>Necessary</td>
</tr>
<tr>
<td>Verify</td>
<td>4000ms</td>
<td>341ms</td>
<td>765ms + Delay</td>
</tr>
</tbody>
</table>

\(^*\) We need to truncate the output of SHA-256 in order to achieve the same computational complexity around \(2^{90}\).

<table>
<thead>
<tr>
<th>Protocol</th>
<th>2048-bit RSA signature</th>
<th>213-bit McSBA ((n = 131072, k = 130885, t = 11))</th>
</tr>
</thead>
<tbody>
<tr>
<td>Operations</td>
<td>RSA Signature + 8(\times) SHA-256</td>
<td>Bit-wise XOR + SHA-256</td>
</tr>
<tr>
<td>Verify</td>
<td>(6.09 \times 10^{-3})s</td>
<td>(4.92 \times 10^{-3})s</td>
</tr>
</tbody>
</table>

C. Established Protocol TESLA

Constructing the one-way hash-chain of verification key \(K_i\) by use of hash function, enables TESLA to generate key dependent on time.

Time delay arises when a sender makes key \(K_i\) open, which makes it hard to authenticate transmitted packets on time. Overall, TESLA makes use of "verification key \(K_i\)”, "MAC key \(K'_i\)”, "MAC" and "time delay", which enjoys the robustness but results in a time delay. Some variants of TESLA may reduce this delay, but McSBA is still faster as we explain in the following. In order to compare TESLA with RSA signature and McSBA under the same security level \(2^{90}\), we need to truncate the output (256-bit) of SHA-256 (SHA-2) operation. Consequently, we set MAC size 180-bit.

1) Applied Hash Function: We employ SHA-2 and HMAC-SHA-2 in order to construct the hash-chain of \(K_i\), and compute a MAC part respectively.

2) Broadcast Time: SHA-2 is used twice at least for both purposes of extracting the past key \(K_i\) and extracting the MAC key \(K'_i\). And HMAC-SHA-2 is used for the purpose of computing the MAC value. Hence, we can define broadcast time as the total airtime of SHA-2 \(\times 2 + \text{HMAC-SHA-2}\).

3) Authentication Time: The authentication is run symmetrically. Hence, it is clear to see authentication time is the total airtime of SHA-2 \(\times 2 + \text{HMAC-SHA-2} + \text{the maximum delay time (Delay)}\). We can say that it takes too long time to verify the transmitted packets.

In addition, each receiver must perform buffering the transmitted packet before he authenticates it. Thus the operation gets tough when a couple of information is burst broadcasted.

D. Comparison Results

In Table 1, we compare the above three schemes roughly. The 2048-bit RSA signature is used as a criterion. We use the...
2^{10} complexity for long-term security. Note that if we adopt 2^{10} security then we can get more efficient performance.

The main operations of the three schemes are listed, where exponentiation is the dominant issue of the RSA signature, and hash computation affects small. But for McSBA and TESLA, hash computation dominates the performance of the schemes. It is because that in McSBA binary matrix operation is very fast, even compared to the hash computation; and in TESLA, hash computation is repeatedly run.

The length of the signature (or MAC) is distinct. At first, RSA has a huge signature, so not desirable in practice. But interestingly, although McSBA uses a signature, its length is similar to that of TESLA which uses MAC.

On the update issue, only TESLA has to refresh the key schedule frequently to keep the hash-chain working. That will consume more power of the devices. On the contrary, using signature for authentication has no such problem, and saves the limited power.

One more property TESLA has is the tight synchronization, which is due to the requirement of robustness. However, it is not so exciting for emergency broadcast, and increases the complexity of the implementation.

Note that TESLA has an essential delay in its robust protocol, which to some extend plays the role of asymmetric cryptography to achieve authentication. “Delay” depends on the time of transmission intervals, which ranges from several seconds to hours and deny the quick response of the protocol. Although one may reduce the “Delay” by other techniques, it will also trade off special properties of TESLA. More importantly, even without “Delay”, TESLA is still slower than McSBA according to our investigation in Table 1.

IV. SIMULATION EVALUATION

In Table 2, we show the results of the simulation about verification in both RSA signature and McSBA. Considering verification time from an implementation aspect, we compare binary matrix operation (McEliece signature) with modulo exponentiation (RSA signature) in simulation. In the same platform, we use Intel(R) Pentium(R) M, processor 1.60GHz, 1.60GB RAM.

A. 2048-bit RSA signature

We monitor the total verification time in the standard OpenSSL. Since a SHA-256 operation can process 256-bit in one time, we have to take at least 8 (= 2048/256) times SHA-256 computation to run 2048-bit RSA signature. Hence, verification time of RSA signature in OpenSSL + 8 × SHA-256 provides the total verification time. As a result, 5.85 × 10^{-4}s + 1.78 × 10^{-5}s = 6.03 × 10^{-4}s. (Though cost of hash computation is negligible compared to RSA exponentiation calculation.)

B. 213-bit McSBA

We compute the binary multiplication s_1 = zH^T and SHA-256 operation as the total verification. Generating a systematic code, it is basically sufficient to compute the (1 × k) matrix × (k × (n-k)) matrix part on s_1 = zH^T. However, in order to evaluate the accurate performance, we consider the computation cost of the whole (1 × n) matrix × (n × (n-k)) matrix part of zH^T. The binary n-bit error vector z contains 11 (= t) errors. In short, “1” exists in only 11 positions. Considering that the information about all error positions are transmitted, we do not require the operation for searching “1” positions in the whole n-bit error vector z.

Hence, we can say that the multiplication of two matrices basically amounts to be only 11 (= t) times of binary exclusive-OR operations corresponding to “1” positions by a row vector. In detail, each 187-bit syndrome generated corresponding to an error position would be partitioned with five 32-bit words, and 11 times of binary exclusive-OR operations are executed per 32-bit word.

Consequently, 66 (= 6(≈ 187/32) × 11) × binary exclusive-OR operation + SHA-256 provides the total verification time in McSBA. Incidentally, in order to reduce the times of SHA-256 operations, assume that the input size of SHA-256 is lower than 447 (= 512 − 1 − 64) bit in this simulation. As a result, 4.7 × 10^{-5}s (a mean value of the hundred times measurement) + 2.22 × 10^{-6}s = 4.92 × 10^{-5}s.

Note that searching t error positions takes relatively long time, in this case. We observed a total 7.52 × 10^{-4}s (a mean value of the hundred times measurement) if without using our modification to McElieic signature, in the same platform. Therefore, in order to reduce verification time, it is good to implement the encoding method introduced in Sec.2.

C. Comparison Results

The obtained results in Table 2, show that McSBA can verify quite faster than RSA signature. Consequently we expect that McSBA has the high potential to verify the authenticity of emergency information in the power-restricted devices instantly.

V. CONCLUSION

In this paper, we reconsider lightweight broadcast authentication protocols. We compare among the cost performances of lightweight authentication broadcast authentication protocols. The lightweight broadcast authentication protocol based on digital signature has not been realized for a long time because computation cost is too high.

By making use of efficiently verifiable McElieic signature, we investigate that the broadcast authentication protocol McSBA can authenticate with a low computation cost applicable on power-restricted devices, faster than 2048-bit RSA signature and the TESLA protocol. Especially, from the estimation results, it is shown that McSBA can verify quickly less than 1s in emergency situations, however RSA signature with the same security takes more than 4s. Our investigation and reference of the implementation results and our simulation show that McSBA is potentially possible to be applied in the low power devices for providing authenticity, due to its fast verification. McSBA might help strengthen the authenticity of not only emergency communication but also wireless sensor network.
networks in a low computation resource. We will further work for experimental performance of McSBA and applications of McEliecie signature.

REFERENCES


APPENDIX

We attach program codes here to prove our simulation results in Sec. 4.

```c
#include <Windows.h>
#include<stdio.h>
#include<stdlib.h>
#include<time.h>
#define WSIZE 32 // wordsize
#define COLS 131072 // n
#define ROWS 187 // n-k
#define DIM 130885 // k = COLS-ROWS
#define WROWS 6 // (n-k)/WSIZE
#define WCOLS 4091 // (COLS-ROWS)/WSIZE
#define NOISE 11 // t

unsigned int h[COLS-ROWS][WROWS];

int main()
{
    int i, j, k;
    unsigned int tmp, quo, rem;
    unsigned int s[WCOLS], zp[NOISE];

    LARGE_INTEGER begin;
    LARGE_INTEGER end;

    LARGE_INTEGER freq;

    // generation of pseudo-public-key
    srand((unsigned int) time(0));
    for(i=0;i<COLS;i++)
        for(j=0;j<WROWS;j++)
            // generation of 32bit random
            tmp=0;
            for(k=0;k<(WSIZE/8)-1;k++)
                tmp+=((unsigned int)
                    (rand()/((double)RAND_MAX+1.0f))*256);
            tmp = tmp << 8;
    for(k=0;k<(WSIZE/8)-1;k++)
        tmp+=((unsigned int)
            (rand()/((double)RAND_MAX+1.0f))*256);

    for(i=0;i<NOISE;i++) {
        zp[i]=(unsigned int)
            (rand()/((double)RAND_MAX+1.0f))*COLS;
    }

    QueryPerformanceFrequency(&freq);
    // Counter Begin
    QueryPerformanceCounter(&begin);
    for(i=0;i<WROWS;i++) s[i]=0;
    for(i=0;i<NOISE;i++)
        if ( zp[i] < ROWS ) {
            quo = zp[i] / WSIZE;
            rem = zp[i] % WSIZE;
            tmp = 1 << rem;
            s[quo] ^= tmp;
        }
        else {
            for(k=0;k<WROWS;k++)
                s[k] ^= h[zp[i]-ROWS][k];
        }
    // Counter End
    QueryPerformanceCounter(&end);

    printf("%f\n", (double)
        (end.QuadPart-begin.QuadPart)
        / freq.QuadPart);
    return (double)
        (end.QuadPart-begin.QuadPart)
        / freq.QuadPart;
}
```

This full text paper was peer reviewed at the direction of IEEE Communications Society subject matter experts for publication in the WCNC 2009 proceedings.