ISSN : 2229-4333(Print) | ISSN : 0976-8491(Online)

Secured Initialization of Dynamic Message Authentication in Wireless Devices

[1]Dr. D. Raghu, [2] Radha Krishna cheepi, [3]Ch. Raja Jacub
[1,2]Nova College of Engg & Tech, Jangareddy gudem.
[3]Dept. of Computer Science, Nova College of Engineering, Jangareddy Gudem

Abstract
There are several protocols and mechanisms have been proposed to address the problem of initial secure key deployment in wireless networks. Most existing approaches work either with a small number of wireless devices (i.e., two) or otherwise rely on the presence of an auxiliary device. On the other hand, such components should support security applications such as message integrity, authentication, and time stamping. The latter are efficiently implemented by Dynamic Message Authentication Codes (DMAC). As clearly stated in the literature, current approved implementations of DMAC require resources that cannot be supported in constrained components. An approach to implement a compact DMAC by the use of stream ciphering is presented in this paper.

Keywords
Secured communications, DMAC, challenge response, ciphers.

I. Introduction

In the Wireless Networks message integrity and authenticity, and replay prevention, are essential in security-related communications. Here, a receiver is expected to be able to verify that a received message, originally transmitted by a valid source, was not changed. Also, the receiver has to verify that the message was not transmitted by a cloned source, and is not a retransmission of an originally genuine message transmitted in the past by a valid source. Technically, verifying message integrity and authenticity is based on the receiver's ability to prove to itself that the transmitter stores a valid secret key that was used when the message was transmitted. Surely, symmetric and asymmetric cryptographic schemes can also be used in satisfying the above. In this paper, we treat the case where the facility at the data source has limited resources. In such environments, message integrity and authenticity is usually verified using Message Authentication Code (MAC). A fundamental cryptographic primitive is that of Message Authentication Codes (MAC), namely, methods for convincing a recipient of a message that the received data is the same that originated from the sender. MACs are extremely important in today's design of secure systems since they reveal to be useful both as atomic components of more complex cryptographic systems and as themselves alone, to guarantee integrity of stored and transmitted data. Traditional message authentication schemes create a hard authenticator, where modifying a single message bit would result in a modification of about half the authentication tag. These MACs fit those applications where the security requirement asks to reject any message that has been altered to the minimal extent. In many other applications, such as those concerning data, there may be certain modifications to the message that may be acceptable to sender and receiver, such as errors in reading data or in communicating passwords through very noisy channels. This new scenario, not captured by the traditional notion of MACs, motivated the introduction and study in [5] of a new cryptographic primitive, a variant of MACs, which was called Dynamic Message Authentication Code (DMAC); namely, methods that propagate "acceptable" modifications to the message to "recognizable" modifications in the authentication tag, and still retain their security against other, unacceptable modifications.

II. MAC and DMAC Context

As described above, MAC(M,K) is a one-way transformation of the message M and a secret key K. The implementation this transformation can be based on various approaches. Dynamic Message Authentication Code (DMAC) is a hash transformation parameterized with a secret key. That is, it is an implementation of MAC(M,K). In this paper, we treat a standardized DMAC, specified in [1-4]. The security of such implementations has been revised [5], stating that the attacks "do not contradict the security proof of DMAC, but they improve our understanding of the security of DMAC based on the existing cryptographic hash functions." The suggested implementation of DMAC is of the form DMAC (text,K)= $H[K_{out}]||H[K_{in}||(\text{text})]$;
where

- H is a cryptographic hash function,
- K_{in} and K_{out} are two keys, derived from K,
- ||denotes a concatenation, and
- text is the text to be hashed together with K. In relation to the challenge response procedure of we adopt an implementation where

text = C||M. The following is one recommended way of constructing K_{in} and K_{out} out of K. Let K be b-bytes long. opad and ipad denote b-byte values, consisting, respectively, of b repetitions of the byte 01011100 and 00110110. Then, K_{out} =K opad and K_{in} = K ipad, where denotes an XOR. Any standard hash (e.g., SHA-1 [6]), as well as the above specified DMAC implementation, is specified in a way which facilitates the processing of a relatively long text, by iteratively processing it in parts. That is, text is broken into sections, which are processed one at a time. Each such section is processed by a one-way block transformation whose parameters are limited in size to that of the processed section. This general approach is especially suitable when dealing with constrained hardware resources. Even if text is a few hundred of bits long, where K is about 100 bits, it would still be recommended to process text in parts.

III. Dynamic MAC

The property that we can require from DMACs is that of preimage-resistance. Informally, we require that the tagging algorithm, if viewed as a function on the message space, is hard to invert, no matter what is the distribution on the message space. (Later, while showing the applications of DMACs to biometric entity authentication, this property will be useful in proving that the entity authentication scheme obtained is secure against adversaries that can gain access to the DMAC output from the biometric storage file.)

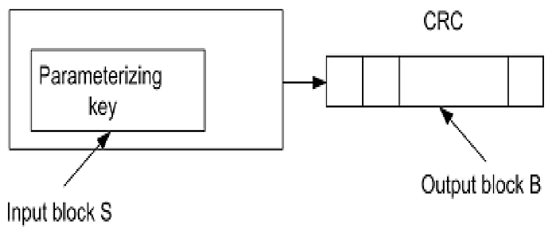
A. Definition

The d_m-ac-as-MAC (K_g,Tag,Verify) is (t, €)-preimage-resistant if the following holds. Let k be generated using K_g; for any algorithm Adv running in time at most t, if Adv queries algorithm Tag (k, ·) with adaptively chosen messages, thus obtaining pairs (m_1,

t_1), . . . , (m_q, t_q), and then returns a message m_0, and is given a value tag =Tag(k,m), the probability that Adv(tag) returns m0 such that Verify (k,m_0 , tag) = 1 and dm(m_0,m_i) ≥ γ for i = 0, 1, . . . , q, is at most €.
We note that essentially all conventional MAC constructions in the literature would satisfy an analogue preimage-resistance requirement. However it is easy to transform a MAC into one that is not preimage-resistant and for some applications like biometric identification, it can be desirable to require that the MAC used is preimage-resistant (or otherwise an accidental loss of the MAC output could reveal a password or some data to an adversary). A second additional property that we can require from DMACs is that of tag public verifiability. Informally, we require that, given two tags obtained from two different messages, it is possible to eficiently verify that the two messages have small distance, without using any secret key. while showing the applications of DMACs to biometric entity authentication, this property will be useful in obtaining a network entity authentication scheme, where the server does not need to run the DMAC to verify tag correctness.

B. Chippers

Cipher is a symmetric encryptor (i.e., the transmitter and receiver share the same secret key). The key forms a seed which generates a pseudorandom keystream. At the transmitting end, this keystream is XOR with the cleartext stream, yielding a ciphertext stream. The receiver, having the same seed key, generates synchronously the same keystream. XORing with the received ciphertext yields the cleartext back. Stream ciphers operate at a higher speed than block ciphers and have relatively low hardware complexity. Fundamental security requirements that should be satisfied by a stream cipher concern randomness characteristics of the generated keystream and inability to recover the secret seed key by knowing the generated keystream. Development and analysis of stream ciphers are subject of continuous activities.
Chipper



C. DMAC Implementation

First of all, we remark that several simple constructions using arbitrary error correcting codes and or dinary MACs fail in satisfying even the approximate correctness and security requirements of DMACs. These include techniques such as interpreting the input message as a codeword, and using a conventional MAC to authenticate its decoding (here, the property of approximate correctness fails). Other techniques that also fail are similar uses of fuzzy commitments from [9], fuzzy sketches from [4] and reusable fuzzy extractors from [2]. We note however that there are a few simple constructions that meet the approximate correctness and security requirements of DMACs but don't meet the preimage-resistance or the efficiency or the tag public variability requirement. The simplest we found goes as follows. Let us denote as (K,T,V) a conventional MAC scheme. The tagging algorithm, on input key k and message m, returns tag = m|T(k,m). The verifying algorithm, on input k,m_0, tag, sets tag = t_1 | t_2 and returns 1 if and only if d(t_1,m_0) ≤ δ and V (k, t_1, t_2) = 1, where d is the distance function. The scheme satisfies the approximate correctness and security, and the tag public verifiability requirements; however, note that the tag of this scheme contains the message itself and therefore the scheme is neither preimage-resistant.

D. DMAC Analysis

Informal description. We explain the ideas behind this scheme in two steps. First, we explain how to use a probabilistic TCR hash function to guarantee that outputs from this hash function will have some additional distance-preserving properties. Second, we show how we can use such probabilistic TCR hash function to construct an approximately correct and secure MAC. We achieve a combination of distance-preserving properties and target collision resistance by making a TCR hash function probabilistic, and using the following technique. First, the message bits are randomly permuted and then the resulting message is written as the concatenation of several equal-size blocks. Here, the size of each block could be the fixed constant size (e.g., 512 bits) of the input to compression functions (e.g., SHA) that are used as atomic components of practical constructions of TCR hash functions. Now multiple hashes are computed, each being obtained using the TCR hash function, using as input the concatenation of a dierent and small enough subset of the input blocks. Here, the choice of each subset is done at random, and specifically, using the output of a random pairwise-independent hash function on input the message. Furthermore, each subset has the same size, depending on the length of the input and on the desired distance-preserving properties. The basic idea so far is that by changing the content of some blocks of the message, we only change a small fraction of the inputs of the atomic hashes and therefore only a small fraction of the outputs of those hashes will change. Given this 'probabilistic TCR hash function', the tagging and verifying algorithm can be described as follows. The tagging algorithm, on input a random key and a message, uses another value, which can be implemented as a counter incremented after each application (or a random value chosen independently at each application). Then the algorithm computes the output of the finite pseudo-random function on input such value and divides this output in two parts: the first part is a random key for the TCR hash function and the second part is a sequence of pseudo-random bits that can be used as randomness for the above described probabilistic TCR hash function. Now, the tagging algorithm can run the latter function to compute multiple hashes of the message. The tag returned is then the input to the finite pseudo-random function and the hashes. The construction of the verifying algorithm is necessarily dierently from the usual approach for exactly correct and secure MACs (where the verifying algorithm runs the tagging algorithm on input the received message and checks that its output is equal to the received tag), as this algorithm needs to accept the same tag for multiple messages. Specifically, on input the tag returned by the tagging algorithm, the verifying algorithm generates a key and pseudo-random bits for the probabilistic TCR hash function exactly as the tagging algorithm does and computes the hashes of the received message. Finally, the verifying algorithm checks that the received and the computed sequences of hashes only dier in a small enough number of positions. Formal description. Let k be a security parameter, t be an approximation parameter, and c be a block size constant. We denote by PIH = {pih | pih : {0, 1}n → {0, 1}n} a set of pairwise independent hash functions over {0, 1} m, by TCRH = {tcrhK : K ε {0, 1}k} a finite TCR hash function, and by F = {fK : K {0, 1}k} a finite pseudo-random function. We now present our construction of an approximately-secure and approximately correct MAC, which we denote as (K_g,Tag,Verify). Instructions for K_g: generate a uniformly distributed k-bit key K

Input to Tag: a k-bit key K, an n-bit message M, parameters p, δ, γ, a block size 1c and a counter ct.
Instructions for Tag:
– Set x1 = n/2δ and x2 = 10 log(1/(1 − p))
– Set (u|π|pih) = fK(ct), where u ε {0, 1}k, π is a permutation of {0, 1}n and pihε PIH
– Write π(M) as M1| · · · |Mdn/ce, where |Mi| = c for i = 1, . . . , dn/ce
– Use pih(M) as randomness to randomly choose x1-size subsets S1, . . . , Sx2 of {1, . . . , dn/ce}
– For i = 1, . . . , x2,
let Ni = Mi1 | · · · |Mix1, where Si = {i1, . . . , ix1}
let shi = tcrhu(Ni)
– Let subtag = sh1| · · · |shx2
– Return: tag = ct|subtag.
– Set ct = ct + 1 and halt.
Input to Verify: parameters δ, γ, a block size 1c , a k-bit key K, an n-bit
message M_0 and a string tag
Instructions for Verify:
– Write tag as ct|u|sh1| · · · |shx2
– Set x1 = n/2δ and x2 = 10 log(1/(1 − p))
– Set (u|π|pih) = fK(ct), where u ε {0, 1}k, π is a permutation of {0, 1}n and pih ∈ PIH
– Write π(M0) as M01| · · · |M0 dn/ce, where |M0 i | = c for i = 1, . . . , dn/ce
– Use pih(M0) to randomly select x1-size subsets S01, . . . , S0 x2 of {1, . . . , dn/ce}
– For i = 1, . . . , x2,let N0 i = M0 i1| · · · |M0ix1, where S0i = {i1, . . . , ix1} let sh0i = tcrhu(N0i)– Check that sh0i = shi, for at least αx2 of the values of i {1, . . . , x2}, for α = 1 − 1/2√e − 1/2e.
– Return: 1 if all verifications were successful and 0 otherwise.
The above construction satisfies the following
Theorem Assume that F is a (tF , qF , F)-secure pseudo-random function and H is a (tH, qH, H)-target collision resistant hash function. Then (Kg,Tag,Verify) is a (p, δ) approximately correct and (dm, γ, tA, qA, ε A)-approximately secureMAC,
where
• dm is the Hamming distance
• γ = 2δ
• A ≤ F + H · qA + 1 − p
• qA = qF ≥ 1 and qH = 10 log(1/(1 − p))
• tA = min(tA,1, tA,2)
• tA,1 = tF −O(qA(n(log n+log(1/(1−p)))+log(1/(1−p))+time(hu; nc/2δ))
• tA,2 = tH − O(n(log n + log(1/(1 − p))) + time(fK; |ct|)) and n is the length of the message, c is a block size constant, ct is the counter input to algorithm Tag, and time(g; x) denotes the time required to compute function g on inputs of size x.

E. Performance

We analyze the main performance parameter of interest that is, the communication complexity of our scheme (Kg,Tag,Verify). We see that the length of the returned tag is x2 · c, where x2 = 10 log(1/(1 − p)), and c is the length of the output of the TCR hash function. We note that c is constant with respect to n, and acceptable settings of parameter p can lie anywhere in the range [1 − 1/2(log n)1+€, 1], for any constant € > 0, and where n is the length of the message input to the scheme. Therefore the length of the tag returned by the scheme can be as small as 10c(log n)1+€ ; most importantly, this holds for any value of parameter δ. The tag length remains much shorter than the message even for much larger settings of p; for instance, if p = 1 − 2−√n, the tag length becomes O(√n).

IV. Conclusion

Wireless SensorNetworks pose a need for efficient implementation of MAC. To achieve efficiency, while not sacrificing security, there is a need to evaluate new approaches, while also utilizing any characteristic of the specific implementation of MAC that can enhance efficiency. A complete highly compact MAC implementation, based on stream ciphering, was presented. The principle was to implement a hash transformation based on the stream cipher, where the strength of the hash is associated with the underlying security of the cipher. The hash is then utilized to implement DMAC, based on standard procedures. A specific implementation, based on DECIM (v2), a highly scrutinized stream cipher, was presented and analyzed in detail.

References

[1] ANS Institution, "Keyed Hash Message Authentication Code", ANSI X9.71, 2000.
[2] National Institute of Standards and Technology, "The Keyed-Hash Message Authentication Code (HMAC)", FIPS PUB 198, Information Technology Laboratory, 2002.
[3] J. Kim, A. Biryukov, B. Preneel, S. Hong, "On the Security of HMAC and NMAC Based on HAVALl, MD4, MD5, SHA-0 and SHA-1", Proc. Conf. Security and Cryptography for Networks (SCN '06), pp. 242-256, 2006.
[4] National Institute of Standards and Technology, "Secure Hash Standard", FIPS PUB 180-1, Information Technology Laboratory, 1995.
[5] GAO, "GAO-05-551 Radio Frequency Identification Technology", [Online] Available : www.gao.gov/new.items/d05551.pdf, May 2005.
[6] European Commission, "Draft Recommendation on RFID Privacy and Security", [Online] Available : http://www.edri.org/edrigram/number6.4/ec-recommandation-rfid, Feb. 2008.
[7] G. Avoine, "RFID Security & Privacy Lounge", [Online] Available : www.avoine.net/ rfid/, 2009.
[8] H. Chan, A. Perrig, D. Song. Random Key Predistribution Schemes for Sensor Networks. In Proceedings of the IEEE Symposium on Security and Privacy, 2003.
[9] C.-H. Owen Chen, C.-W. Chen, C. Kuo, Y.-H. Lai, J. M. McCune, A. Studer, A. Perrig, B.-Y. Yang, T.-C. Wu. GAnGS: Gather, Authenticate 'n Group Securely. In Proceedings of the 14th Annual International Conference on Mobile Computing and Networking, MO- BICOM, 2008
[10] W. Du, J. Deng, Y. S. Han, P. K. Varshney. A Pairwise Key Pre-Distribution Scheme for Wireless Sensor Networks. In ACM CCS, 2003.