Evaluation of Some Algorithms for Hardware-Oriented Message Authentication

Ågren, Martin; Hell, Martin; Johansson, Thomas

2011

Evaluation of Some Algorithms for Hardware-Oriented Message Authentication

Martin Ågren, Martin Hell, and Thomas Johansson

Dept. of Electrical and Information Technology, Lund University,
P.O. Box 118, 221 00 Lund, Sweden

Abstract. In this technical report, we consider ultra light-weight constructions of message authentication in hardware applications. We examine several known constructions and evaluate details around their hardware implementations. These constructions are all based on the framework of universal hash functions.

Keywords: Message Authentication Codes, Universal Hash Functions, Hardware Implementation, Lightweight, Stream Cipher

1 Introduction

Message Authentication Codes (MAC:s) [6] is a class of keyed functions used in many different areas to ensure that a message has been sent by the true sender and received without having been altered during transmission. The sender produces a tag $\boldsymbol{t} = f(\boldsymbol{k}_{\text{MAC}}, \boldsymbol{m})$ for a message $\boldsymbol{m}$ using a key $\boldsymbol{k}_{\text{MAC}}$ and attaches it to the message. The receiver, who shares the key $\boldsymbol{k}_{\text{MAC}}$, can produce the tag for the received message and immediately decide whether the message can be regarded as authentic.

An active adversary tries to modify a transmitted message and its tag and hopes to get this accepted at the receiver side. We would like the probability that the attacker succeeds to be some very small value.

There are many ways to provide message authentication, e.g., using certain block cipher modes of operation, using a keyed hash function, or using constructions based on universal hashing [15, 14]. This report focuses on the last approach, which is the usual choice when we assume that encryption is done through a stream cipher (unless there is some dedicated authentication inside the stream cipher). Typical examples would be the GCM mode [11] and the UIA2 [1] found in UMTS and LTE.

One major new application area in cryptology is light-weight cryptology. Many recent applications put very strong demands on parameters of cryptographic algorithms. In particular, we have in mind passive RFID applications, where the total gate count must be kept very low. No acceptable solutions using symmetric algorithms have yet been demonstrated in practice. Recently, a new RFID authentication protocol based on a stream cipher has been presented [4].

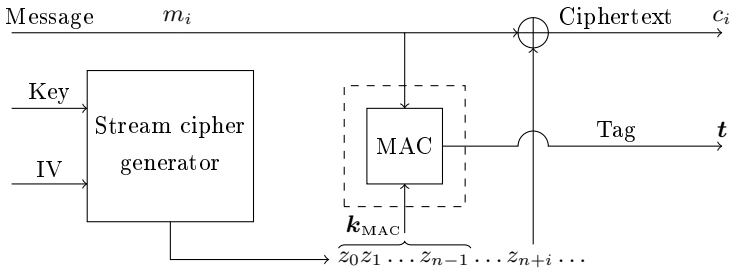

Fig. 1. The setting we consider in this paper. The first n output bits from a stream cipher generator, initialized with a key and IV, form $\boldsymbol{k}_{\text{MAC}}$ which is used to initialize the authentication mechanism. The rest of the stream cipher bits are used to encrypt the message bits m_i, $i = 0, 1, \ldots$. The end result is a sequence of ciphertext bits c_i and the authentication tag $\boldsymbol{t}$.

Motivated by this, we consider ultra light-weight constructions of message authentication in hardware applications like RFID.

We examine several known constructions and evaluate details around their hardware implementations. More specifically, we try to assess the cost of a simple implementation which uses no lookup tables or advanced techniques. We quantify the cost in terms of registers and gates.

2 Preliminaries

We are interested in a mechanism combining encryption and authentication in some packet-based communication system, or similar. We assume that the encryption is done through a modern stream cipher, using a secret key $\boldsymbol{k}$ and a public initial value (IV). The stream cipher would either be a dedicated one, or a suitable mode of operation for a block cipher. The key $\boldsymbol{k}_{\text{MAC}}$ used to produce the MAC is derived from $\boldsymbol{k}$ by using keystream before encryption starts.

For security reasons, we have to extract the full $\boldsymbol{k}_{\text{MAC}}$ before encryption, even though part of $\boldsymbol{k}_{\text{MAC}}$ is typically not used until the very end of the algorithm, after encryption has completed. The details on why one cannot extract randomness after encryption has finished is available in [2].

Summarizing this, we present an overview of this approach in Fig. 1. This is an adopted and standard approach, used in GCM, GSM, UMTS, etc. The remainder of this report will focus on the dashed rectangle in Fig. 1 which inputs a key and a message and outputs a tag. (One might also authenticate the cipherbits c_i instead of message bits m_i.)

When we study known constructions, we ignore the process of creating key bits $\boldsymbol{k}_{\text{MAC}}$. We consider the hardware cost for the MAC generation part, as marked in Fig. 1. For some constructions, we have detailed a hardware implementation and will mention the gate count obtained. When assessing the costs

Table 1. The gate counts used for different functions.

Flip flop	8
NAND2	1
XOR2	2.5
MUX2	4
MUX3	5
DEMUX3	5

for certain hardware primitives (gates), we have used cost figures found in several other papers, e.g., [7]; they are found in Table 1. We typically assume a tag size of 64 bits.

2.1 MAC:s and Universal Hash Functions

In order to fully appreciate the theory behind the constructions in this report, we need to understand "universal hash functions". An overview of this theory is available in e.g., [2]. As we only study details surrounding the implementations, we do not go into the exact details on why these particular algorithms are secure.

Without going into the precise definitions, let us note that the constructions we study will be either ϵ-almost universal (ϵ-AU) or ϵ-almost XOR universal (ϵ-AXU). Here, ϵ can be directly translated to a success probability for an attack, and thus we want to keep ϵ "small".

When constructing a MAC using an ϵ-AU family H, the key $\boldsymbol{k}_{\text{MAC}}$ is divided into two parts. One part selects a particular function $h \in H$, and the other is used with a block cipher to encrypt the output of h. On the other hand, with an ϵ-AXU family, the final encryption can be done using a simple XOR with the other part of the key (a one-time pad). This class of families is interesting, since a stream cipher is often to prefer over a block cipher in the settings we consider.

2.2 Notation

We use $||$ to denote concatenation of bitstrings. 0^w denotes a bitstring consisting of w zeroes. The length of m, encoded using w bits, is denoted $[m]_w$.

3 A Hardware Evaluation of Existing Constructions

We revisit some known constructions. The first three use polynomial evaluation. The basic idea is to combine message blocks and key blocks. A common approach is to try to avoid multiplications in favour of additions.

The last three constructions we look at are all ϵ-AXU. Notably, they only update what will be the output using XOR, meaning that the one-time pad can be added in the very beginning just as well as at the very end. We note that this allows us to save one register.

3.1 GCM: GMAC

$w \geq 128$ [11] is the block length and H is a hash-key. Ignoring what [11] calls "additional authenticated data", the m-bit message is split into w-bit blocks m_i, $1 \leq i \leq n-1$. We set $m_n = 0^{w/2}||[m]_{w/2}$ and let

$$\text{GHASH}(H, m, m_n) = \sum_{i=1}^{n} m_i H^{n+1-i}$$

be the internal keyed hash where all calculations are performed in some fixed representation of $\mathbb{F}_{2^w}$. The tag is then encrypted using a key (related to H) and a value J derived from the IV. [11, Appendix A] shows that GHASH is $n2^{-t}$-AXU when $n-1$ is the maximum allowed message block count and t the length of the truncated output.

Interpreting the formula for GHASH, we view the message blocks as coefficients of a polynomial which we evaluate in the point H. An implementation would consist of five w-bit registers as in Fig. 2 where the next message block is loaded in parallell with processing the current one. The final XOR-ing of random bits cannot be performed ahead of time, which means that a special register is needed for keeping this value until the very end of the process.

The bottom register contains the value H and the message is loaded bit by bit into the message buffer. Immediately prior to loading a message block m_i from the message buffer, the Z register contains the tag accumulated thus far. $m_i \oplus Z$ is then put into the V register and the Z register is reset. By clocking the two lower registers and updating the Z register with the contents of V iff the left-most bit of the lower register is a one, the multiplication with H is calculated. After w such clockings, the lower register is back at its starting value (H). At this point, everything is prepared for the immediate loading of another message block m_{i+1}. The final tag is $Z \oplus OTP$.

A rough calculation of the hardware cost, $5w$ flip flops, w AND:s, w XOR:s and a small number of multiplexers, yields 2800 gates. To save on one register, we could get rid of the message buffer at the cost of lowering the speed to half. This would give a gate count around 2100, but this figure cannot be compared to those later in this report, since the performance is inferior.

3.2 UMTS: UIA2

There are large similarities with GCM, but also some differences [1]: Ignoring the differing block sizes in the original specifications, one difference that emerges is that the padding is done by choosing $m_n = [m]_w$. We need two random values (apart from the final OTP): P (w bits) and Q (w bits). We then calculate

$$\text{Hash} = \text{trunc}_{w/2}(\text{GHASH}(P, m, m_n) \cdot Q)$$

which is the evaluation of the "polynomial" m in the point P, multiply with Q and truncate half the bits. A performance difference, comparing to GMAC,

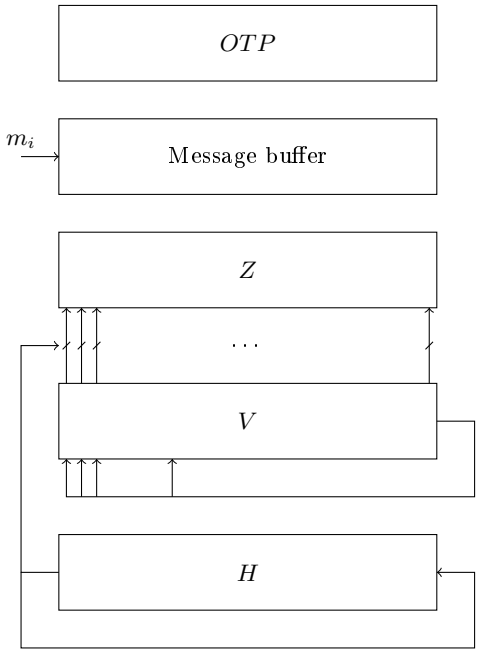

Fig. 2. A hardware implementation of GHASH.

emerges when we study the effect of the multiplicating with Q and truncating: we cannot move the use of Q before the processing of the message, meaning that we need to store this value somewhere during the entire process. This increases our cost in terms of registers compared to the GMAC-construction.

Why are these extra operations added to the result of GHASH then? While the GHASH-call is a $n2^{-w}$-AXU hash family, the final multiplication and truncation is $2^{-w/2}$-AXU. This makes $(n2^{-w}+2^{-w/2})$-AXU combined. Comparing this to the GHASH-construction, the difference emerges for e.g., $n = 2^{w/2}, t = w/2$. The GMAC construction achieves 1-AXU, while UIA2 clocks in at a less laughable security of $2^{1-w/2}$.

Studying these two constructions, it seems one protects longer messages (much) better at the cost of some additional post-processing which unfortunately cannot be implemented in a manner which suits us.

3.3 WH

This [8] is a modification of a scheme named NH used in UMAC [5]. While the latter uses integer additions, the construction studied here works with polynomials over $\mathbb{F}_2$, meaning that a hardware implementation does not need to carry

during addition. The function is defined as

$$\mathrm{WH}_k(m) = \sum_{i=1}^{n/2} (k_{2i-1} \oplus m_{2i-1}) \cdot (k_{2i} \oplus m_{2i}) x^{(\frac{n}{2}-i)w} \bmod p$$

where $p = p(x)$ is a polynomial of degree w irreducible over $\mathbb{F}_2$. This function is 2^{-w}-AU on equal-length strings using $w \geq 1$ and even $n \geq 2$. Comparing to NH, which is also 2^{-w}-AU on such strings, a key difference is that that the tags are of different lengths, where WH produces tags whose "security" actually match their size.

The problem with this design is that it is AU, not AXU. This difference is crucial since it is not enough to mask the tag with output from the stream cipher generator. Using block ciphers solves this problem at an apparent cost, which is not acceptable to us.

3.4 Cryptographic CRC

Krawczyk [9] suggests using a CRC with a secret irreducible polynomial $p(x)$ and calculating the tag as

$$h_p(m) = m(x) \cdot x^n \bmod p(x).$$

Note that the message-tag pair (m, h) is accepted iff $p(x)|q(x)$ where

$$q(x) = m(x) \cdot x^n - h(x).$$

An implementation is outlined in Fig. 3. The top register contains the OTP, added during finalization. Message bits are shifted into the middle register. When the bit shifted out of the middle register is a one, the contents of the lower register is added to the middle one and the left-most bit in the middle register is flipped (this represents the "+1" in the feedback polynomial).

Using $3w$ flip flops, w XOR:s, $2w$ AND:s and a small number of multiplexers, we have arrived at a gate count around 1830. One problem is that the feedback polynomial must be chosen randomly, yet it has to be irreducible. This is a major drawback.

3.5 LH and UH

Sarkar [13] writes about a construction which he claims can be implemented very efficiently using a word-oriented LFSR. Furthermore, he suggests an application using a stream cipher to generate a keystream, some of which is used for encryption and some of which is used for authentication.

Example 1. This is a simplified version of an example in [13] where we avoid word-oriented LFRS:s and use "ordinary" LFSR:s. We use a full-period LFSR of length w and a w-bit tag accumulator and start by loading key material into the

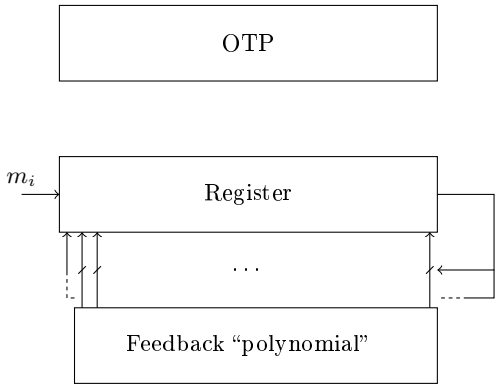

Fig. 3. A hardware implementation of the CRC construction.

shift register. For each bit to authenticate, we either XOR the accumulator with shift register content or we don't, depending on whether the authenticated bit is 1 or 0, respectively. At each step, we also clock the LFSR. After authenticating w bits, we load new key material and proceed in the same fashion.

From a security standpoint, it is tempting to use the above example, but the keystream consumption will be very large and, in fact, the authentication will use just as much keystream as the encryption.

3.6 An LFSR-Based Toeplitz Construction

We start by filling an LFSR $\boldsymbol{K}$ with key material and choosing a secret irreducible feedback polynomial. At each time step, this LFSR is clocked and we denote the content at time i as $\boldsymbol{K}_i$. Maintain a w-bit state $\boldsymbol{t} = t_0 \dots t_{w-1}$ initialized with zeroes.

For each bit m_i we wish to authenticate, if it is zero we do nothing and if it is one we update $\boldsymbol{t} \leftarrow \boldsymbol{t} \oplus \boldsymbol{K}_i$. We may write this as

$$\boldsymbol{t} = \begin{bmatrix} t_0 \\ t_1 \\ \vdots \\ t_{w-1} \end{bmatrix} = \begin{bmatrix} k_{-1} & k_0 & k_1 & \dots & k_{L-2} \\ k_{-2} & k_{-1} & k_0 & \dots & k_{L-3} \\ \vdots & \vdots & \vdots & \dots & \vdots \\ k_{-w} & k_{1-w} & k_{2-w} & \dots & k_{L-1-w} \end{bmatrix} \begin{bmatrix} m_0 \\ m_1 \\ m_2 \\ \vdots \\ m_{L-1} \end{bmatrix}$$

to authenticate L message-bits m_i.

It can be shown, by following [3, 12, 10], that this family of hash functions is ϵ-AXU where $\epsilon = \frac{L}{2^{w-1}}$ (see also [9]). That is, the security degrades with the maximum allowed length of the messages. Note that this result is stated incorrectly in [10]: there is a factor two missing. See [2] for details.

Please also note the similarities with Example 1 above. In that construction, we maintained a higher security at the price of constantly refilling the state of the

LFSR. The price then was keystream consumption; here, there is a cost which is equally unwanted: we need to randomly choose the irreducible polynomial.

4 Conclusion

In this technical report, we have seen how the lowest number of w-bit registers needed is three. All of the hardware-efficient algorithms turned out not to be particularly suitable for our setting as they rely on one or more of

- a block cipher to do post-processing,
- multiplication in some field,
- special randomness, e.g., an irreducible polynomial,
- as many pseudorandom bits as message bits.

Acknowledgment

This work was supported by the Swedish Foundation for Strategic Research (SSF) through its Strategic Center for High Speed Wireless Communication at Lund.

References

1. 3GPP. Specification of the 3GPP Confidentiality and Integrity Algorithms UEA2 & UIA2; Document 1: UEA2 and UIA2 specifications. TS 35.215, 3rd Generation Partnership Project (3GPP), September 2006.
2. M. Ågren, M. Hell, and T. Johansson. On hardware-oriented message authentication with applications towards RFID. In *Proceedings of the 2011 Workshop on Lightweight Security & Privacy: Devices, Protocols, and Applications*. IEEE Computer Society Conference Publishing Services, 2011.
3. N. Alon, O. Goldreich, J. Håstad, and R. Peralta. Simple construction of almost k-wise independent random variables. *Annual IEEE Symposium on Foundations of Computer Science*, pages 544–553, 1990.
4. O. Billet, J. Etrog, and H. Gilbert. Lightweight privacy preserving authentication for RFID using a stream cipher. In S. Hong and T. Iwata, editors, *Fast Software Encryption 2010*, volume 6147 of *Lecture Notes in Computer Science*, pages 55–78. Springer-Verlag, 2010.
5. J. Black, S. Halevi, H. Krawczyk, T. Krovetz, and P. Rogaway. UMAC: Fast and secure message authentication. In *Advances in Cryptology—CRYPTO'99*, pages 215–233. Springer-Verlag, 1999.
6. E. N. Gilbert, F. J. MacWilliams, and N. J. A. Sloane. Codes which detect deception. *Bell Systems Technical Journal*, 53(3):405–424, 1974.
7. M. Hell, T. Johansson, and W. Meier. Grain - a stream cipher for constrained environments. *International Journal of Wireless and Mobile Computing, Special Issue on Security of Computer Network and Mobile Systems.*, 2(1):86–93, 2006.
8. J.-P. Kaps, K. Yüksel, and B. Sunar. Energy scalable universal hashing. *IEEE Transactions on Computers*, 54(12):1484–1495, 2005.

9. H. Krawczyk. LFSR-based hashing and authentication. In *Advances in Cryptology—CRYPTO'94*, pages 129–139. Springer-Verlag, 1994.
10. H. Krawczyk. New hash functions for message authentication. In *Advances in Cryptology—EUROCRYPT'95*, pages 301–310. Springer-Verlag, 1995.
11. D. A. McGrew and J. Viega. The security and performance of the Galois/Counter Mode (GCM) of operation. In A. Canteaut and K. Viswanathan, editors, *Progress in Cryptology—INDOCRYPT 2004*, volume 3348 of *Lecture Notes in Computer Science*, pages 343–355. Springer-Verlag, 2004.
12. J. Naor and M. Naor. Small-bias probability spaces: Efficient constructions and applications. *SIAM Journal on Computing*, 22(4):838–856, 1993.
13. P. Sarkar. A new universal hash function and other cryptographic algorithms suitable for resource constrained devices. Cryptology ePrint Archive, Report 2008/216, 2008. *http://eprint.iacr.org/*.
14. D. R. Stinson. Universal hashing and authentication codes. In J. Feigenbaum, editor, *Advances in Cryptology—CRYPTO'91*, volume 576 of *Lecture Notes in Computer Science*, pages 74–85. Springer-Verlag, 1992.
15. M. N. Wegman and J. L. Carter. New hash functions and their use in authentication and set equality. *Journal of Computer and System Sciences*, 22:265–279, 1981.