Energy Scalable Universal Hashing

Jens-Peter Kaps, *Student Member, IEEE,* Kaan Yüksel, *Student Member, IEEE,* and Berk Sunar, *Member, IEEE*

***Abstract*— Message Authentication Codes (MACs) are valuable tools for ensuring the integrity of messages. MACs may be built around a universal hash function (NH) which was explored in the construction of UMAC. In this paper, we use a variation on NH called WH. WH reaches optimality in the sense that it is universal with half the hash length of NH and it achieves perfect serialization in hardware implementation. We achieved substantial power savings of up to 59% and a speedup of up to 7.4 times over NH. Moreover, we show how the technique of multi-hashing and the Toeplitz approach can be combined to reduce the power and energy consumption even further while maintaining the same security level with a very slight increase in the amount of the key material. At low frequencies the power and energy reductions are achieved simultaneously while keeping the hashing time constant. We developed formulae for estimation of the leakage and dynamic power consumptions as well as the energy consumption based on the frequency and the Toeplitz parameter t. We introduce a powerful method for scaling WH according to specific energy and power consumption requirements. Our implementation of WH-16 consumes only $2.95\,\mu$W at 500 kHz. It can therefore be integrated into a self-powered device.**

***Index Terms*— Universal hashing, provable security, message authentication codes, scalability, low-power, low-energy.**

I. Introduction

COMPUTING technology is reaching every corner of our lives. Mobile communication, personal computation (e.g. personal digital assistants: PDAs), and portable navigation devices are just a few examples of the most commonly known applications. Recent advances in ultra-low-power technology enabled the development of even smaller, more mobile, autonomous devices. Piconet [1], RFIDs [2], and "Smart Dust" [3] are a few examples of this trend. Common to all these devices is that they communicate wirelessly and their energy source is extremely limited. Batteries for these devices are tiny and can supply $10\,\mu$W for only one day [3]. Moreover, some of these technologies collect energy from environmental sources, such as light, heat, noise, or vibration using *power scavengers*. Scavengers based on micro-electromechanical systems (MEMS) produce around $8\,\mu$W [4] relying solely on ambient vibration. A major application of this technology is distributed sensor networks.

Protecting the integrity of data is of utmost importance for many application scenarios. For example, smart dust motes that are embedded in a bridge could monitor the stress and inform the authorities in case of emergency. Wireless sensors might monitor plant growth, moisture and PH-value on a farm. In both cases the data is not confidential but its authenticity and integrity are very important. For this purpose, efficient *Message Authentication Codes* (MACs) [5] may be preferable over digital signature schemes [6] due to their high encryption throughput and short authentication tags. A disadvantage for both digital signature schemes and traditional MACs is that they provide only computational security. This means that an attacker with sufficient computational power may break the scheme. More severely, the lack of a formal security proof makes these schemes vulnerable to possible shortcut attacks.

Universal hash functions, first introduced by Carter and Wegman [7], provide a unique solution to the aforementioned security problems. Roughly speaking, universal hash functions are collections of hash functions that map messages into short output strings such that the collision probability of any given pair of messages is small. A universal hash function family can be used to build an unconditionally secure MAC. For this, the communicating parties share a secret and randomly chosen hash function from the universal hash function family, and a secret encryption key. A message is authenticated by hashing it with the shared secret hash function and then encrypting the resulting hash using the key. Carter and Wegman [8] showed that when the hash function family is strongly universal, i.e. a stronger version of universal hash functions where messages are mapped into their images in a pairwise independent manner, and the encryption is realized by a one-time pad, the adversary cannot forge the message with probability better than that obtained by choosing a random string for the MAC.

Black et al. [9] describe a new, provably secure message authentication code (UMAC), which has been designed to achieve extreme speeds in software implementations. A hash function family named NH underlies hashing in UMAC. In this paper we improve upon NH in order to make secure hash functions possible in ultra-low-power devices. We implement NH with power efficiency guidelines in mind and notice that its power consumption exceeds our limits by far. Instead of optimizing the implementation even more and reducing its power consumption by a fraction we take a different approach. We identify the main power consumers (i.e. registers, adders) and carefully remove components one by one. We formulate the resulting new algorithm (WH) mathematically and prove that it is still at least as secure as NH. While WH is consuming an order of magnitude less power than NH, its leakage power consumption remains a bottleneck. The leakage power is proportional to the circuit size which is proportional to the size of the hash value which in turn is proportional to the security level. The technique of multi-hashing was introduced [10] to increase the security level of a given hash function without changing the size of the hash value at the expense of more key material. We reverse this procedure to preserve the security

This material is based upon work supported by the National Science Foundation under Grants No. ANI-0112889 and ANI-0133297.

J.-P. Kaps and K. Yuksel are with the Worcester Polytechnic Institute, Electrical and Computer Engineering Department, 100 Institute Rd., Worcester, MA 01609. E-mail: kaps@wpi.edu and kyuksel@wpi.edu.

B. Sunar is with the Worcester Polytechnic Institute, Atwater Kent Room 216, 100 Institute Rd., Worcester, MA 01609. E-mail: sunar@wpi.edu.

level while reducing the size of the hash value and therefore the leakage power. We use the Toeplitz approach to reduce the amount of key material needed. The resulting design is scalable and can be tailored to specific energy and power consumption requirements without sacrificing security.

The remainder of this paper is organized as follows. In Section IV we introduce WH, a power optimized version of NH, with an emphasis on hardware implementation. In Section V we rigorously prove that WH is universal. Section VI describes how we can further reduce the power consumption while maintaining the same level of security by employing the technique of multi-hashing in combination with the Toeplitz approach. We describe our implementation of both NH and WH algorithms in Section VII and present the results in Section VIII. Furthermore, we explain how the Toeplitz parameter t can be used to optimize WH with respect to specific energy and power consumption requirements.

II. Previous Work

The security aspects of distributed sensor networks have been reviewed by *NAI Labs* in [11]. However, this study focused only on software implementations on current processors whose energy consumption is far above the amount that can be supplied by a scavenger circuit. To our knowledge not much work has been done on improving the performance of universal hashing in hardware. Ramakrishna published a study on the performance of hashing functions in hardware based on universal hashing [12]. However, the main emphasis was on using hash functions for table organization and address translation. In an early work Krawczyk [13] proposed new hash functions from a hardware point of view. This work introduced two constructions: a CRC-based cryptographic hash function, and a construction based on Toeplitz hashing where matrix entries are generated by a Linear Shift Feedback Register (LFSR). The reference gives a sketch for hardware implementation, which includes a key spreader. However, it is difficult to estimate the power consumption of this function from a sketch. There have been no implementations reported so far. In the past decade we have seen many new hash constructions being proposed, constantly improving in speed and collision probability [9], [14]–[18]. For a survey see [19]. However, most of these constructions have targeted efficiency in software implementations, with particular emphasis on matching the instruction set architecture of a particular processor or taking advantage of special instructions made available for multimedia data processing (e.g. Intel's MMX technology). While such high end platforms are essential for everyday computing and communications, in numerous embedded applications (e.g. PDAs, mobile phones) space and power limitations prohibit their employment.

III. Preliminaries

A. Notations

Let $\{0,1\}^*$ represent all binary strings, including the empty string. The set $H = \{H_K : A \rightarrow B\}$, is a family of hash functions with domain $A \subseteq \{0,1\}^*$ of size a and range $B \subseteq \{0,1\}^*$ of size b. H_K denotes a single hash function chosen from the set of hash functions H according to a random key $K \in C$ where the set $C \subseteq \{0,1\}^*$ denotes the finite set of key strings. In the text we will set $h = H_K$ for convenience.

The element $M \in A$ stands for a message string to be hashed and is partitioned into blocks as $M = (m_1, \cdots, m_n)$, where n is the number of message blocks of length w. Similarly the key $K \in C$ is partitioned as $K = (k_1, \cdots, k_n)$, where each block k_i has length w. We use the notation $H[n, w]$ to refer to a hash function family where n is the number of message (or key) blocks and w is the number of bits per block.

Let U_w represent the set of nonnegative integers less than 2^w, and P_w represent the set of polynomials over $GF(2)$ [20] of degree less than w. Note that each message block m_i and key block k_i belongs to either U_w, P_w or $GF(2^w)$. Here $GF(2^w)$ denotes the finite field of 2^w elements defined by $GF(2)[x]/(p)$, where p is an irreducible polynomial of degree w over $GF(2)$. In this setting the bits of a message or key block are associated with the coefficients of a polynomial. To illustrate, suppose $w = 6$ and $p = x^6 + x + 1$. Let us see how two messages (binary bit strings), 101101 and 100011, can be multiplied in the Galois Field of $GF(2)[x]/(p)$. 101101 and 100011 would be mapped into $x^5+x^3+x^2+1$ and x^5+x+1, respectively. Multiplication of these two polynomials yields $x^{10} + x^8 + x^7 + x^6 + 2x^5 + x^4 + 2x^3 + x^2 + x + 1$. This is equivalent to $x^{10} + x^8 + x^7 + x^6 + x^4 + x^2 + x + 1$ (since $2x^5 \equiv 0x^5 \equiv 0$ in $GF(2)$). Dividing this polynomial by p and taking the remainder, we obtain $x^5 + x^3 + x^2 + x$ (corresponding to the bit string 101110). Note that this way carries are eliminated as well. Finally the addition symbol '+' is used to denote both integer and polynomial addition (in a ring or finite field). The meaning should be obvious from the context.

B. Universal Hashing

A universal hash function, as proposed by Carter and Wegman [7], is a mapping from the finite set A with size a to the finite set B with size b. For a given hash function $h \in H$ and for a message pair (M, M') where $M \neq M'$ the following function is defined: $\delta_h(M, M') = 1$ if $h(M) = h(M')$, and 0 otherwise, that is, the function δ yields 1 when the input message pairs collide. For a given finite set of hash functions $\delta_H(M, M')$ is defined as $\sum_{h \in H} \delta_h(M, M')$, which tells us that $\delta_h(M, M')$ yields the number of functions in H for which M and M' collide. When h is randomly chosen from H and two distinct messages M and M' are given as input, the collision probability is equal to $\delta_h(M, M')/|H|$. We give the definitions of the two classes of universal hash functions used in this paper from [19]:

Definition 1: The set of hash functions $H = h : A \rightarrow B$ is said to be **universal** if for every $M, M' \in A$ where $M \neq M'$,

$$|h \in H : h(M) = h(M')| \ = \delta_H(M, M') = \frac{|H|}{b} \ .$$

Definition 2: The set of hash functions $H = h : A \rightarrow B$ is said to be ϵ**-almost universal** ($\epsilon - AU$) if for every $M, M' \in A$ where $M \neq M'$,

$$|h \in H : h(M) = h(M')| \ = \delta_H(M, M') = \epsilon |H| \ .$$

In this definition ϵ is the upper bound for the probability of collision. Observe that the previous definition might actually be considered as a special case of the latter with ϵ being equal to $1/b$. The smallest possible value for ϵ is $(a-b)/(b(a-1))$.

In the past many universal and almost universal hash families were proposed [9], [14]–[18]. Black et al introduced an almost universal hash function family called NH in [9]. The definition of NH is given below.

Definition 3: ([9]) Given $M = (m_1, \cdots, m_n)$ and $K = (k_1, \cdots, k_n)$, where m_i and $k_i \in U_w$, and for any even $n \geq 2$, NH is computed as follows:

$$\mathsf{NH}_K(M) = \left[\sum_{i=1}^{n/2} ((m_{2i-1} + k_{2i-1}) \bmod 2^w) \cdot ((m_{2i} + k_{2i}) \bmod 2^w) \right] \bmod 2^{2w} .$$

In the same paper NH was shown to have a tight bound of 2^{-w} on the collision probability.

IV. Weighted NH-Polynomial with Reduction (WH)

We introduce a new hash function family WH, as a variation to the NH construction, which improves upon NH in terms of power, area and speed. The monomial $x^{(\frac{n}{2}-i)w}$, which is constant irrespective of the input, serves to optimize the hardware implementation (see Section VII-B).

Definition 4: Given $M = (m_1, \cdots, m_n)$ and $K = (k_1, \cdots, k_n)$, where m_i and $k_i \in GF(2^w)$, for any even $n \geq 2$, and a polynomial p of degree w irreducible over $GF(2)$, WH is defined as follows:

$$\mathsf{WH}_K(M) = \sum_{i=1}^{n/2} (m_{2i-1} + k_{2i-1}) \cdot (m_{2i} + k_{2i})\, x^{(\frac{n}{2}-i)w} \pmod p .$$

In this construction message and key blocks are associated with polynomials over $GF(2)$ as opposed to their integer counterparts in the NH construction. In a hardware implementation this completely eliminates the carry chain and thereby improves all three efficiency metrics (i.e. speed, space, power) simultaneously: Due to the elimination of carry propagations, the operable clock frequency (and thus the speed of the hash algorithm) is dramatically increased. Likewise, the area efficiency is improved since the carry network is eliminated. Finally, due to the reduced switching activity, the power consumption is reduced.

Moreover, the new scheme provides us with shorter authentication tags. The size of the authentication tag is a concern for two reasons. First, the tag needs to be transmitted along with the data. Therefore, the shorter the tag, the less power will be consumed for transmission purposes. Second, the size of the tag determines the number of flip-flops needed for storing the tag. NH requires a large number of flip-flops for the double length hash output. In this construction, the storage and transmission requirements are improved by introducing a reduction polynomial of degree matching the block size, and hence, reducing the size of the authentication tag by half. Note that one of the main motives [9] of the NH construction was to eliminate the modular reductions used in the previously proposed hash families (e.g. MMH proposed in [15], SQUARE proposed in [18]) since reductions are relatively costly to implement in software. A modulo reduction involves division and computation of the remainder. In hardware, however, reductions (especially those with fixed low-weight polynomials) can be implemented quite efficiently. The reduction becomes an integral part of the computation and involves only a simple subtraction at each step (see Section VII-B).

Another point is that while processing multiple blocks, it is often necessary to hold the hash value accumulated during the previous iterations in a temporary register. This increases the storage requirement and translates into a larger and slower circuit with higher power consumption. We could achieve perfect serialization via scaling each product of message and key pairs with a power of x. In the implementation this translates into the accumulation of block products in the same register that holds the hash of the previously processed blocks. This eliminates the need for an extra temporary register as well as other control components required to implement the data path.

V. Analysis

In this section we give the theorem and its proof establishing the security of WH.

Theorem 1: For any even $n \geq 2$ and $w \geq 1$, $\mathsf{WH}[n, w]$ is universal on n equal-length strings.

The intuition behind this theorem is that when WH is used as the hash function, we can mathematically prove and quantify that the adversary cannot falsify our message with a better probability than randomly selecting the right hash value from all possible hashes.

Proof: Let M, M' be distinct members of the domain A with equal lengths. For brevity we denote $(m_{2i-1} + k_{2i-1})(m_{2i} + k_{2i}) = mk_{2i}$, $(m'_{2i-1} + k_{2i-1})(m'_{2i} + k_{2i}) = m'k_{2i}$ and so on. Let M, M' be distinct members of the domain A with equal lengths. We are required to show that

$$Pr\left[\mathsf{WH}_K(M) = \mathsf{WH}_K(M')\right] = 2^{-w} .$$

Expanding the terms inside the probability expression, we obtain

$$Pr\left[\sum_{i=1}^{n/2} mk_{2i} \left(x^{(\frac{n}{2}-i)w}\right) = \sum_{i=1}^{n/2} (m'k_{2i} \left(x^{(\frac{n}{2}-i)w}\right) \pmod p \right] = 2^{-w} . \quad (1)$$

The probability is taken over uniform choices of $(k_1, \ldots, k_n)$ with each $k_i \in GF(2^w)$ and the arithmetic is over $GF(2^w)$. Since M and M' are distinct, $m_i \neq m'_i$ for some $1 \leq i \leq n$. Let $m_{2l} \neq m'_{2l}$. For any choice of $k_1, \ldots, k_{2l-2}, k_{2l}, \ldots, k_n$

having

$$Pr_{k_{2l-1}\in GF(2^w)}\left[\sum_{i=1}^{n/2} mk_{2i}\left(x^{(\frac{n}{2}-i)w}\right) = \sum_{i=1}^{n/2} m'k_{2i}\left(x^{(\frac{n}{2}-i)w}\right) \pmod p\right] = 2^{-w} \quad (2)$$

satisfied for all $1 \le l \le n/2$ implies (1). Setting y and z as

$$y = \left[\sum_{i=1}^{l-1} m'k_{2i}x^{(\frac{n}{2}-i)w} - \sum_{i=1}^{l-1} mk_{2i}x^{(\frac{n}{2}-i)w}\right] \pmod p$$

and

$$z = \left[\sum_{i=l+1}^{n/2} m'k_{2i}x^{(\frac{n}{2}-i)w} - \sum_{i=l+1}^{n/2} mk_{2i}x^{(\frac{n}{2}-i)w}\right] \pmod p$$

we rewrite the probability expression in (2) as

$$Pr_{k_{2l-1}}\left[x^{(\frac{n}{2}-l)w}\left[mk_{2l} - m'k_{2l}\right] = y + z \pmod p\right] = 2^{-w}\ .$$

Since $x^{(\frac{n}{2}-l)w}$ is invertible in $GF(2^w)$, the equation inside the probability expression can be rewritten as follows.

$$k_{2l-1}(m_{2l} - m'_{2l}) + m_{2l-1}(m_{2l} + k_{2l}) - m'_{2l-1}(m'_{2l} + k_{2l}) = x^{-(\frac{n}{2}-l)w}(y+z) \pmod p$$

Solving the equation for k_{2l-1}, we end up with the following

$$k_{2l-1} = (m_{2l} - m'_{2l})^{-1}\Big((x^{-(\frac{n}{2}-l)w})(y+z) - m_{2l-1}(m_{2l} + k_{2l}) + m'_{2l-1}(m'_{2l} + k_{2l})\Big) \pmod p\ .$$

Note that $(m_{2l} - m'_{2l})$ is invertible since in the beginning of the proof we assumed that $m_{2l} \neq m'_{2l}$. This proves that for any m_{2l}, m'_{2l} (with $m_{2l} \neq m'_{2l}$) and y, $z \in GF(2^w)$ there exists exactly one $k_{2l-1} \in GF(2^w)$ which causes a collision. Therefore,

$$Pr\left[\mathsf{WH}_K(M) = \mathsf{WH}_K(M')\right] = 2^{-w}\ .$$

■

VI. Reducing the Power Consumption with Toeplitz Construction

The power consumed by a VLSI circuit has two components: Leakage power and dynamic power. Only the latter depends on frequency. This means that at lower frequencies the total power consumption is dominated by the leakage power consumption. Since this component is directly proportional to the size of the circuit, we now aim to design a smaller circuit, and hence, introduce the hash function family $\mathsf{WH}^T[n, w, t]$ ("Toeplitz-WH") having three parameters, namely n, w and t. The additional parameter t stands for Toeplitz iteration count, where $t \ge 1$, and the others are defined as before. Domain A remains the same whereas the range is now $B = \{0,1\}^{wt}$. A function is selected by a key K of length $w(n+2(t-1))$ bits. In other words, K is composed of $(n+2(t-1))$ w bit words. We have $K = (k_1, k_2 \cdots, k_{n+2(t-1)})$, where each k_i is a w bit word. The notation $K_{i..j}$ represents $K = (k_i, k_{i+1}, \cdots, k_j)$. Then for a message string $M \in A$, $\mathsf{WH}^T_K(M)$ is defined as follows.

$$\mathsf{WH}^T_K(M) = (\mathsf{WH}_{K_{1..n}}(M), \mathsf{WH}_{K_{3..n+2}}(M), \cdots, \mathsf{WH}_{K_{2t-1..n+2t-2}}(M)).$$

The circuit size scales with the data path width, i.e. the block size w of the message and the key. Since the collision probability is equal to 2^{-w} (see Section V), reducing the block size w will significantly increase this probability and impair the security of the system. In order to decrease the collision probability without changing the word size, [9] uses the technique of multi-hashing [10] in which different random members of the hash function family are applied to the message, and the results are concatenated to form the hash value. We use a similar approach, however, we preserve the collision probability while reducing the word size. For instance, to obtain the collision probability of 2^{-w} with a block size of $w/4$ bits, each message block is hashed 4 times with independent keys. The computed hash outputs ($w/4$ bits each) are then concatenated to form the w bit hash result. The drawback of this method is that it requires 4 times the key material. As a remedy one can employ the well-known Toeplitz approach [9], [13], [21] in which shifted versions of one key rather than several independent keys are used. In this case, however, since the keys are related to each other, it is not obvious that the collision probability can be maintained. In Theorem 2 we prove that the Toeplitz construction for WH can still achieve the desired result.

Theorem 2: For any $w \ge 1$, $t \ge 1$, and any even $n \ge 2$, $\mathsf{WH}^T[n, w, t]$ is universal on equal-length strings with collision probability of 2^{-wt}.

Proof: For the sake of brevity we will use WH and WH^T instead of $\mathsf{WH}[n, w]$ and $\mathsf{WH}^T[n, w, t]$, respectively. We denote $(k_{2i+2j-3} + m_{2i-1})(k_{2i+2j-2} + m_{2i}) = km_{2i,2j}$, $(k_{2i+2j-3} + m'_{2i-1})(k_{2i+2j-2} + m'_{2i}) = km'_{2i,2j}$ and so on. Let M and M' be distinct members of the domain A with equal lengths. We are required to show

$$Pr[\mathsf{WH}^T_K(M) = \mathsf{WH}^T_K(M')] = 2^{-wt} \quad (3)$$

We have $M = (m_1, m_2, \cdots, m_n)$, $M' = (m'_1, m'_2, \cdots, m'_n)$ and $K = (k_1, k_2, \cdots, k_{n+2(t-1)})$, where m_i, m'_i and k_i are all w bit words associated with polynomials. Note that the arithmetic is carried out over $GF(2^w)$ with the irreducible polynomial p of degree w. Next we define the event E_j for $j \in \{1, \cdots, t\}$ as follows.

$$E_j : \sum_{i=1}^{n/2} km_{2i,2j}\left(x^{(\frac{n}{2}-i)w}\right) = \sum_{i=1}^{n/2} km'_{2i,2j}\left(x^{(\frac{n}{2}-i)w}\right) \pmod p$$

We call each term in the summations of the E_j a "clause" (e.g., $(k_1+m_1)(k_2+m_2)x^{(\frac{n}{2}-1)w}$ is a clause). Now the probability in (3) can be rewritten as

$$Pr[E_1 \cap E_2 \cap \cdots \cap E_t] \ .$$

Without loss of generality, we can assume that M and M' disagree in the last clause (i.e., $m_{n-1} \neq m'_{n-1}$ or $m_n \neq m'_n$). Notice that if M and M' agreed in the last clause then each E_j would be satisfied if and only if it was also satisfied when that last clause was omitted. Hence, we could truncate M and M' after the last clause in which they disagree, and still obtain exactly the same set of keys causing collisions.

Now, again without loss of generality, we can assume that $m_{n-1} \neq m'_{n-1}$ because for each iteration of E_j the key is shifted by two words making the case symmetric. We proceed by proving that for all $j \in \{1, \cdots, t\}$, $Pr[E_j \ is \ true \ | \ E_1, \cdots, E_{j-1} \ are \ true] = 2^{-w}$, implying the theorem.

For $j = 1$, the claim is satisfied due to Theorem 1. For $j > 1$, the events E_1 through E_{j-1} depend only on key words $k_1, \cdots, k_{n+2j-4}$ while E_j depends also on k_{n+2j-3} and k_{n+2j-2}. By fixing k_1 through k_{n+2j-4} such that E_1 through E_{j-1} are satisfied, and fixing any value for k_{n+2j-3}, we prove that there is only one value of k_{n+2j-2} satisfying E_j. Let

$$y = \sum_{i=1}^{n/2-1} km'_{2i,2j}\left(x^{(\frac{n}{2}-i)w}\right) - \sum_{i=1}^{n/2-1} km_{2i,2j}\left(x^{(\frac{n}{2}-i)w}\right) \ .$$

Thus, E_j becomes

$$E_j : km_{n,2j} - km'_{n,2j} = y \pmod p \ .$$

Now we are required to prove that

$$Pr[km_{n,2j} - km'_{n,2j} = y \pmod p] = 2^{-w} \ .$$

Solving the equation inside the above probability expression for k_{n+2j-2}, we end up with the following

$$k_{n+2j-2} = (m_{n-1} - m'_{n-1})^{-1}\left(y - m_n(m_{n-1} + k_{n+2j-3}) + m'_n(m'_{n-1} + k_{n+2j-3})\right) \pmod p \ .$$

Note that $(m_{n-1} - m'_{n-1})$ is invertible since in the beginning of the proof we assumed $m_{n-1} \neq m'_{n-1}$. This proves that for any k_{n+2j-3}, m_{n-1}, m'_{n-1} (with $m_{n-1} \neq m'_{n-1}$) $\in GF(2^w)$ there exists exactly one $k_{n+2j-2} \in GF(2^w)$ which causes a collision. Therefore,

$$Pr[\mathsf{WH}_K^T(M) = \mathsf{WH}_K^T(M')] = 2^{-wt} \ .$$

■

VII. Implementation

The power dissipation in CMOS devices can be characterized by the following formula [22]:

$$P = \underbrace{\left(\frac{1}{2} \cdot C \cdot V_{DD}^2 + Q_{se} \cdot V_{DD}\right) \cdot f \cdot N}_{\mathrm{P_{Dynamic}}} + \underbrace{I_{leak} \cdot V_{DD}}_{\mathrm{P_{Leakage}}} \quad (4)$$

The term $\mathrm{P_{Dynamic}}$ represents the power required to charge and discharge circuit nodes as well as the power dissipation during output transitions. The terms C, Q_{se}, and V_{DD} are technology dependent [22]. The switching activity, i.e. the number of gate output transitions per clock cycle, is represented by N, and the operating frequency by f. The second term $\mathrm{P_{Leakage}}$ represents the static power dissipation due to the leakage current I_{leak}. The leakage current is directly determined by the number of gates and the fabrication technology. For more information about low-power design see [23]. In order to minimize the power consumption, we designed our CMOS circuits with the following rules in mind:

- The number of output transitions has to be minimal.
- The circuit size should be minimized.
- Glitches cause unnecessary transitions and therefore should be avoided.

A. NH

The algorithm for NH is described in [9]. It is given in this paper in Definition 3 as

$$\mathsf{NH}_K(M) = \left[\sum_{i=1}^{n/2}((m_{2i-1} + k_{2i-1}) \bmod 2^w) \cdot ((m_{2i} + k_{2i}) \bmod 2^w)\right] \bmod 2^{2w} \ .$$

This leads to the simple functional block diagram shown in Figure 1. The message and the key are assumed to be split into n blocks of w bits. Messages that are shorter than a multiple of $2 \cdot w$ are padded. All odd message blocks are applied to input $m1$, all even message blocks to input $m2$. The blocks of the key are applied similarly to $k1$ and $k2$. The output of Adder 1 is $ma = m1 + k1 \bmod 2^w$, the output of Adder 2 is $mb = m2 + k2 \bmod 2^w$. These are integer additions where the carry out is discarded. The multiplication results in $mout = ma \cdot mb$. The final adder accumulates all $n/2$ products.

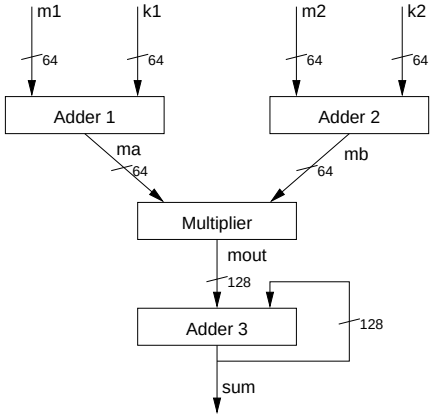

Fig. 1. Functional diagram for NH

The actual block diagram for the circuit is much more complex and can be found in Figure 2. As power consumption is our main concern and not speed, we base our design on a bit serial multiplier. For each multiplication of two w bit

numbers, w partial products need to be computed and added: $mout = \sum_{j=1}^{w} ma \cdot mb[j] \cdot 2^{j-1}$.

This decision gives us the ability to use a bit serial adder for Adder 2 as its result $mb[j]$ (indicated as $mult$ in Figure 2) can directly be used by the bit serial multiplier. A bit serial adder produces one bit of the result with each clock cycle, starting with the LSB, and it has minimal glitching. However, the multiplicand ma has to be available immediately. Therefore we use a simple ripple carry adder to implement Adder 1. Its main disadvantage is that it takes a long time until the carries propagate through the adder, causing a lot of glitching and therefore a high power consumption. However, Adder 1 needs to compute a new result only every 64 clock cycles, hence its dynamic power consumption is tolerable.

The Bit Multiplier in Figure 2 computes the partial products, one during each clock cycle. The addition of the partial products is accomplished using a carry-save adder and the Right Shift Algorithm [24]. This adder uses the redundant carry-save notation which results in minimal glitching as the carries are not fully propagated. However, this requires 64 additional flip-flops to store the carry bits. After one multiplication has been computed, its result has to be added to the accumulation of the previous multiplications as indicated by Adder 3 in Figure 1. Rather than having a separate multiplier and adder, in the actual implementation we add the partial products of the next multiplication immediately to the result of the previous additions. This technique stores the result of Adder 3 in the Multiplier thus saving a 128 bit register and a 128 bit multiplexer.

The carry-save adder has separate data paths for sum and carry. It can add the partial products of one multiplication very efficiently. However, after the product is computed it needs to be re-aligned before the partial products of the next multiplication can be added to this result. This re-alignment involves converting the number from carry-save notation to standard binary notation, i.e, adding the caries to the sum. This addition is done using a ripple carry adder (Figure 2 shows that this Ripple Carry Adder has the signal rcasum as output). Even though the products of the multiplication are 128 bits wide, the carry is only 64 bits wide, hence the ripple carry adder is only 64 bits wide. This sum needs to be computed only after a multiplication has finished, i.e., every 64 clock cycles. As the result is not needed during the other 63 clock cycles, we isolate the operands from the ripple carry adder, hence the adder does not consume power due to switching activity when its output is not needed. After one multiplication is completed and the result is re-aligned, the carry registers are set to zero for the next computation.

B. WH

The design of WH was conceived by inspecting our implementation of NH and removing the main power consumers. The main drawback of NH for hardware implementations is its use of integer arithmetic. Even though we use a carry-save adder in the multiplier, merge Adder 3, and use ripple carry adders with operand isolation on paths that are active only every 64 clock cycles, the integer adders still consume most of the power. Furthermore, the carry-save adder uses an extra set of flip-flops to store the carries. The delay of the ripple carry adder for Adder 1 makes another set of flip-flops necessary. The bit serial adder stores both inputs in registers for shifting. All these blocks add up to a significant amount of dynamic and leakage power consumption. Our goal for designing WH is to eliminate most of these power consumers.

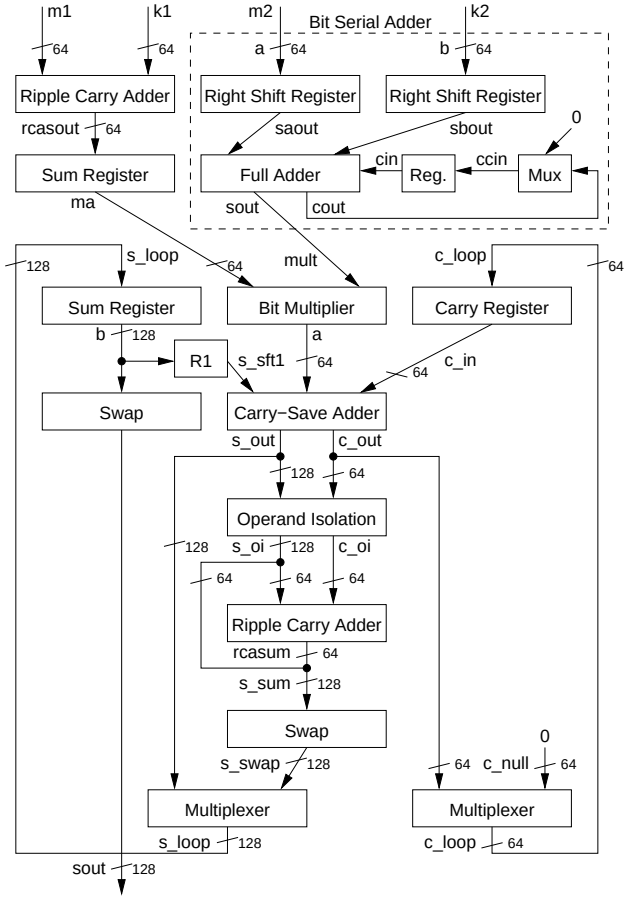

Fig. 2. Block diagram for NH datapath

The first step is to use polynomials over $GF(2)$ instead of integers. Using such polynomials completely removes the carries. Therefore the costly adders can be replaced with simple XOR gates which consume significantly less dynamic and leakage power. Because of their much shorter delay we do not need a register after Adder 1. Also the implementation of the bit serial adder (Adder 2) becomes much simpler, only one 64 bit shift register is needed. We also replaced the carry-save adder of the Multiplier and Adder 3 combination with XOR gates, and thereby removed the whole carry path including the carry register, the multiplexer, and the ripple carry adder to realign the sum.

Moreover, we are reducing the result to 64 bits using an irreducible polynomial. This provides a shorter authentication tag, reducing the power consumed for its transmission. In our hardware implementation the iterations of the multiplication and the reduction operations are interleaved eliminating the need for extra space to store the partial product. Furthermore, using low Hamming-weight polynomials the reduction can

be achieved with only a few gates and minimal extra delay. We are performing the modulo reduction after every single addition. This keeps the reduction circuit simple and the result is never larger than w bits. However, the Multiplier and Adder 3 can no longer be merged due to the reduction. This is expensive as we need to have not just one adder but also one extra 64 bit register. In order to avoid this penalty we multiply each product by $x^{(\frac{n}{2}-i)w}$. This enables us to merge the Multiplier and Adder 3, eliminating the need for the register and the adder. These modifications lead to Definition 4 for WH:

$$\mathsf{WH}_K(M) = \sum_{i=1}^{n/2} (m_{2i-1} + k_{2i-1}) \cdot (m_{2i} + k_{2i})\, x^{(\frac{n}{2}-i)w} \pmod{p}\ .$$

Note that even though the functional diagram for WH (Figure 3) differs from the one for NH (Figure 1) only in the size of the output and the modulo reduction block, the block diagram of WH (Figure 4) differs vastly from the one for NH (Figure 2).

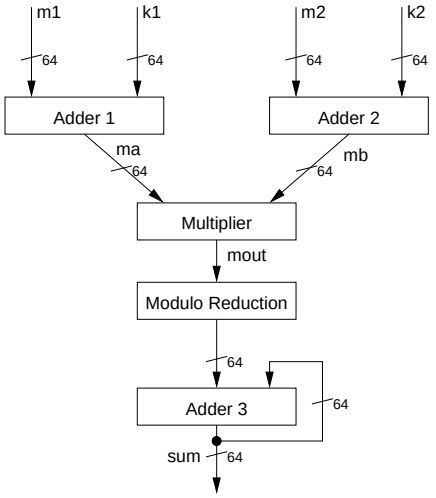

Fig. 3. Functional diagram for WH

C. WH with Toeplitz Construction

We have shown in Section VI that it is possible to preserve the security level while reducing the word size if the message is hashed multiple times with independent keys. The Toeplitz approach describes how these keys can be generated efficiently. For our implementation we assume that the circuit, which generates the messages and the keys, implements this approach and delivers keys and the appropriate parts of the message to our hash function implementation.

Figure 4 shows a detailed block diagram for WH depending on the Toeplitz parameter t. We define the word size w as 64 bits. The block size is the word size divided by the Toeplitz parameter t. The implementation of WH with a 64 bit word size, i.e. $t = 1$, is called WH-64. The minimum input length in this case is $2 \cdot w = 128$ bits. Half of these bits are applied to m1 and the other half to m2. The same holds for the key. In order to achieve the same level of security for a word size of 32 bits we would hash the message twice. Hence, the Toeplitz iteration count t would be two. The implementation of this is called WH-32. In order to hash the same input of 128 bits WH-32 would need to compute four hashes. The length of the final output is the same.

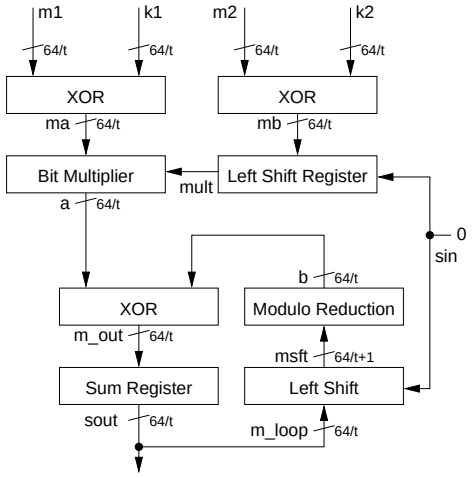

Fig. 4. Detailed block diagram for WH datapath depending on Toeplitz parameter t

D. Control Logic

The control logic manages the switching of the multiplexers, loading of the next data set and resetting of the carry registers. Due to the iterative nature of our multiplier, the control logic requires a counter. Traditionally, counters are built using a register and a combinational incrementer. The incrementer requires long carry propagations which cause glitching and introduce latency. As the critical delay of the datapath is minimized in our design to only a few levels of logic, the delay of an incrementer would create a bottleneck in the control circuit. Hence, optimization of this unit is critical. Instead of an integer counter, we use a linear feedback shift register (LFSR) with 6 flip-flops for NH and WH-64, enhanced to "count" up to 64. LFSRs have minimal glitching and therefore make power efficient and fast counters. The control logic for WH-32 and WH-16 uses the same principle and "counts" only to 32 and 16. The same control logic was used for NH and all versions of WH.

VIII. Results

We used the TSMC $0.13\,\mu\mathrm{m}$ ASIC library, which is characterized for power, and the Synopsys tools Design Compiler and Power Compiler for synthesizing our designs. The results of the simulations on many input sets were verified with the Maple package [25] for consistency.

A. NH and WH

Table I shows the results for power, area, and delay for the hash implementations of NH and WH, synthesized for 100 MHz. WH consumes 41% of the dynamic power and

TABLE I
COMPARISON OF NH WITH WH AT 100 MHZ

Design	Dynamic Power		Leakage Power		Energy		Circuit Area		Delay / Speedup	
	μW	%	μW	%	nJ	%	gates	%	ns	x
NH	1093.9	100	28.1	100	0.72	100	5291	100	9.92	1.0
WH	452.3	41	9.4	33	0.30	41	1701	32	1.35	7.4

33% of the leakage power of NH while at the same time consuming only 32% of the area[1]. WH and NH need the same number of clock cycles to compute a hash value from the same input but WH can run 7.4 times faster than NH. We proved that WH provides the same level of security.

B. WH with Various Block Sizes

We implemented WH with block size w of 64 bits (WH-64), 32 bits (WH-32), and 16 bits (WH-16). Table II shows the results for power, area, and delay for these hash implementations, synthesized for operation at 100 MHz .

TABLE II
COMPARISON OF HASH IMPLEMENTATIONS AT 100 MHZ

Design	Dynamic Power		Leakage Power		Circuit Area		Delay / Speedup	
	μW	%	μW	%	gates	%	ns	x
WH-64	452.3	100	9.36	100	1701	100	1.35	1.0
WH-32	217.5	48	4.81	51	873	51	1.31	1.0
WH-16	126.2	28	2.32	25	460	27	0.76	1.8

It can be seen in Table II that the dynamic and leakage power consumptions as well as the circuit size are reduced almost linearly with the block size. We analytically verify these observations. For simplicity, in our analysis we ignore the contributions of the control and reduction units to the power consumption. From the power dissipation formula for CMOS (Equation 4) we see that the leakage power is proportional to the number of gates (i.e. area A) used: $P_{Leak} \propto A$. The area in turn is proportional to the block size, i.e. $A \propto w$, and therefore

$$P_{Leak} \propto w \ .$$

The dynamic power consumption is proportional to the operating frequency and the number of logic transitions: $P_{Dyn} \propto f\,N$. Since $N \propto w$, the dynamic power consumption scales with the frequency and the block size as follows.

$$P_{Dyn} \propto f\,w$$

The total power consumption $P = P_{Dyn} + P_{Leak}$ is related to f and w as

$$P \propto w(cf + 1) \ .$$

Here c is a fixed constant factor. The energy E consumed is the total power times the running time: $E = PT$. Since $T = \frac{w}{f}$, the total energy consumption is related to the block size and the frequency as

$$E \propto w^2 \left(c + \frac{1}{f}\right) \ .$$

The slight nonlinearity observed in Table II can be explained by considering the control and the modulo reduction units, which are the only parts in the circuit that do not scale linearly with the block size. The size of the modulo reduction unit depends on the primitive polynomial and can be made negligible by utilizing a low-Hamming weight polynomial such as a trinomial. The control unit scales with the logarithm of the block since an LFSR of r flip flops may be used to count through $2^r - 1$ states. This explains why the reduction is not exactly linear. The critical timing path in all implementations is from the control logic to the shift register.

C. WH with Toeplitz

Table III shows the power consumptions of three implementations of WH. The first one is the standard implementation of WH with a block size of $w = 64$. The other two implementations are utilizing the multi-hashing technique with $t = 2$ and 4, and with block sizes of $w = 16$ and 32, respectively. The figures given in Table III represent the power/energy consumptions of the three hash algorithms for processing the same amount of input data (i.e. 64 bits).

In the table we observe that both the dynamic and the leakage power consumptions decrease almost linearly with increasing multi-hash iteration count t. We observe the same behavior for all frequencies. On the other hand the energy consumption stays about the same regardless of multi-hashing and only increases with decreasing operating frequency. Also notice that the leakage power remains the same and it becomes the limiting factor at lower frequencies. One way to reduce the dynamic power consumption is to lower the operating frequency. However, this increases the energy consumption as the leakage power is now consumed over a longer period of time.

As evident from the table using the Toeplitz approach it is possible to reduce the power consumed to hash w bits of data. We next analyze the dependency of power and energy on the block size, the operating frequency, and the multi-hashing iteration count. As a first step we define w as a constant block size of 64 bits. The Toeplitz count is t. In order to achieve the same security for an implementation with a block size of $\frac{w}{t}$ the result has to be hashed t times. The effective block length becomes $w' = \frac{w}{t}$. This approach reduces the power

[1]The area is given in terms of two input equivalent NAND gates.

TABLE III
COMPARISON OF POWER AND ENERGY CONSUMPTION

Design	100 MHz				500 kHz				1 kHz			
	P_{Dyn} μW	P_{Leak} μW	P μW	E nJ	P_{Dyn} μW	P_{Leak} μW	P μW	E nJ	P_{Dyn} nW	P_{Leak} μW	P μW	E nJ
WH-64	452.3	9.36	461.7	0.30	2.261	9.36	11.62	1.49	4.523	9.36	9.37	599.5
WH-32	217.5	4.81	222.3	0.28	1.087	4.81	5.90	1.51	2.175	4.81	4.82	616.4
WH-16	126.2	2.32	128.6	0.33	0.631	2.32	2.95	1.51	1.262	2.32	2.31	592.9

consumed to hash a block of w bits independently of the operating frequency as

$$P'_{Dyn} \propto f\,w' = f\frac{w}{t}\ , \text{ and}$$
$$P'_{Leak} \propto w' = \frac{w}{t}\ .$$

The total power consumption is found as

$$P' \propto \frac{w}{t}(cf+1)$$

where c is a fixed constant factor. This is in line with what we have observed in Table III: The total power consumption is reduced by a factor of t. This improvement does not come for free. Since we have now t blocks of length $\frac{w}{t}$, where each will be hashed t times, it will take t times longer to compute the hash of w bits of data: $T' = t\,T = t\frac{w}{f}$. However, the energy remains unaffected:

$$E' = P'\,T' \propto w^2\left(c+\frac{1}{f}\right)\ .$$

Figure 5 shows how the power consumption of a circuit depends on its area and the clock speed. The graph is extrapolated from simulation data at 100 MHz. It shows clearly that at low frequencies the power consumption scales linearly with the area, i.e. the leakage power is the dominant part. At higher frequencies the dynamic power takes over. The dynamic power consumption scales linearly with the frequency. Note that the frequency axis in Figure 5 is logarithmic and only the powers of ten are shown.

The energy consumption is shown in Figure 6. The axes have a different orientation than in Figure 5 such that the frequency is decreasing towards the right and the area is decreasing towards the left. The frequency axis in Figure 6 is in logarithmic scale. Figure 6 demonstrates that the energy consumption decreases linearly with increasing frequency. However, the energy consumption is independent of the area. This allows us to reduce the circuit size, i.e. increase the Toeplitz parameter t, without any penalty on the energy consumption. Reducing the circuit size decreases the leakage power and at low frequencies this has a big impact as shown in Figure 5. It is now possible to increase the frequency to a level such that the power consumption is the same as it was before reducing the area. Looking back into Figure 6, we can see that the energy consumption is reduced while the power consumption remained the same. This is a powerful tool for optimizing this hash function with respect to specific energy and power consumption requirements.

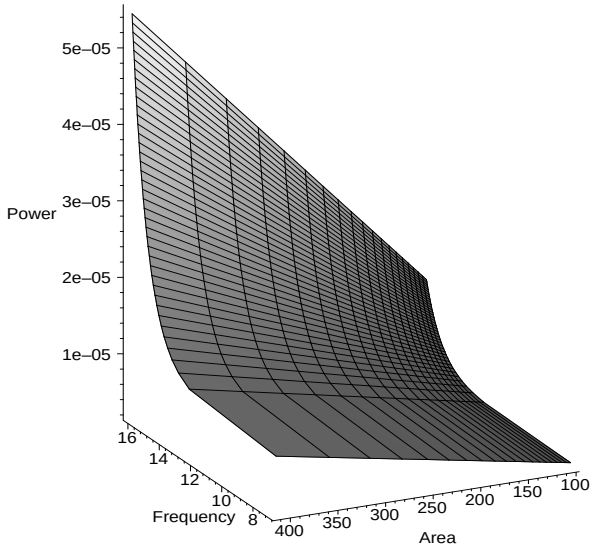

Fig. 5. Power Consumption

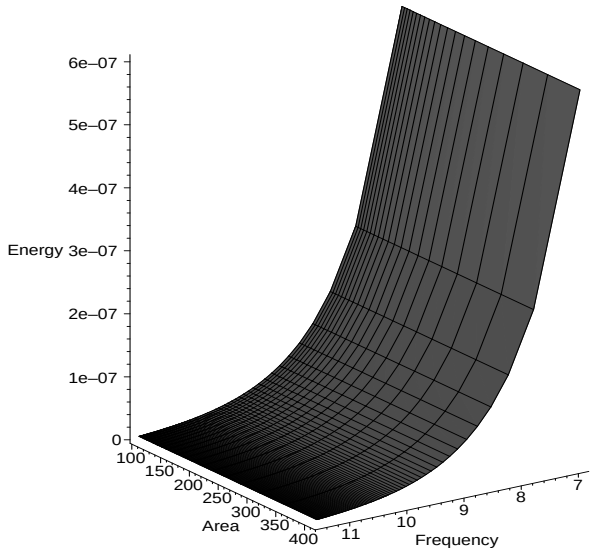

Fig. 6. Energy Consumption

a) Equalizing Runtime: We have demonstrated that the Toeplitz construction provides a drastic t-fold reduction in power consumption and circuit size at the price of t-times slower hash computation. In order to maintain the runtime one may decide to increase the operating frequency t times: $f'' = f\,t$. In this arrangement the dynamic power consumption does not depend on t anymore, only the leakage power does.

TABLE IV
COMPARISON OF POWER AND ENERGY CONSUMPTION WITH $f' = f\,t$

Design	f =100 MHz					f =500 kHz					f =1 kHz				
	f' MHz	P_{Dyn} μW	P_{Leak} μW	P μW	E nJ	f' kHz	P_{Dyn} μW	P_{Leak} μW	P μW	E nJ	f' kHz	P_{Dyn} nW	P_{Leak} μW	P μW	E nJ
WH-64	100	452.3	9.36	461.7	0.30	500	2.261	9.36	11.62	1.49	1	4.523	9.36	9.37	599.5
WH-32	200	435.5	4.81	440.0	0.28	1000	2.175	4.81	7.00	0.89	2	4.350	4.81	4.82	308.4
WH-16	400	505.0	2.32	507.3	0.32	2000	2.525	2.32	4.84	0.62	4	5.050	2.32	2.32	148.5

$$f'' = f\,t \qquad T'' = T \propto \frac{w}{f}$$

$$P''_{Dyn} \propto f''\,w' = f\,w \qquad P''_{Leak} = P'_{Leak} \propto \frac{w}{t}$$

$$P'' \propto w\left(cf + \frac{1}{t}\right) \qquad E'' \propto w^2\left(c + \frac{1}{t\,f}\right)$$

The most important result of this is that at low frequencies (i.e. $P''_{Dyn} \ll P''_{Leak}$) the total power consumption as well as the energy consumption scales with the Toeplitz parameter t.

$$\text{for low frequencies}: E'' \propto \frac{1}{t}\frac{w^2}{f} \qquad P'' \propto \frac{1}{t}w$$

$$\text{for high frequencies}: E'' \propto w^2 \qquad P'' \propto w\,f$$

Table IV shows that the energy needed to compute the hash of a 128 bit data block can be reduced without affecting the runtime. The dynamic power consumption remains roughly constant as time increases, but the leakage power consumption is reduced. Note that the header of the table specifies the frequency f only. The actual clock frequency f' for WH-64 is equal to f, for WH-32 it is twice higher ($t = 2$) and for WH-16 it is four times higher ($t = 4$).

The only way to reduce the leakage power of a circuit, aside from using a different technology, is to reduce the circuit size. Multiple hashing enables us to reduce the circuit size while maintaining the security level. The amount of additional key material is reduced through the Toeplitz approach so that this becomes a viable solution. Table IV shows that at 500 kHz we can reduce the power and energy consumptions by more than half and still compute the hash with the same security and in the same amount of time. This frequency is used in sensor node implementations [26]. WH can operate with as little as $4.8\,\mu$W at 2000 kHz. This is in the range of the power produced by a MEMS scavenger [4]. We would like to note that we used an ASIC standard cell library to obtain these results. A full custom IC-design would yield even higher power savings. We also think that further research could be done to apply the technique of voltage scaling to both algorithms to improve the power savings.

IX. CONCLUSION

In this paper, we propose a variation on NH (the underlying hash function of UMAC), namely WH. Our main motivation was to prove that universal hash functions can be employed to provide provable security in low-power applications. More specifically, we considered hardware implementations of universal hash functions with an emphasis on low-power and reasonable execution speed.

NH produces a hash of length $2w$ and was shown to be 2^{-w}-almost universal. On the other hand, WH reaches optimality and is proven to be a universal hash function family with a much shorter hash length of w. Since its combinatorial properties are mathematically proven, there is no need for making cryptographic hardness assumptions or using a safety margin in practical implementations. In addition, this scheme is simple enough to allow for efficient constructions.

Our implementation of WH shows power savings of up to 59% for dynamic power and 67% for leakage power consumption. It uses 68% fewer gates and can run 7.4 times faster than NH. However, we observed that at lower operating frequencies, the leakage power constitutes the dominant part of the overall power consumption. The only way to reduce the leakage power is to reduce the circuit size. Therefore, we applied multi-hashing integrated with the Toeplitz approach to our hash function WH resulting in the designs WH-32 and WH-16. Without sacrificing any security we achieved drastic power savings of up to 90% over NH and reduced the circuit size by more than 90% to less than 500 gates at the expense of a very slight increase in the amount of key material.

We presented a powerful method for optimizing WH with respect to specific energy and power consumption requirements. We have shown that with the introduction of multi-hashing (t times) together with the Toeplitz approach the circuit size and the power consumption is reduced by a factor of t while it takes t times longer to compute the hash. Therefore the energy consumption stays about the same. On the other hand the operating frequency may be increased t times to achieve the hash without increasing the runtime. Then the dynamic power consumption is increased t-fold, however, the leakage power is not affected. Hence, at low frequencies (i.e. $P_{Dyn} \ll P_{Leak}$) the total power consumption as well as the energy consumptions decrease linearly with increasing parameter t. This is a powerful technique to decrease the circuit size, and the power and energy consumptions simultaneously while maintaining the hashing speed. The only limiting factor is the maximum operating frequency.

By designing the new algorithms with efficiency guidelines in mind and applying optimization techniques, we achieved drastic power, energy and area savings. Our implementation of WH-16 consumes only $2.95\,\mu$W at 500 kHz and uses only 460 gates. It could therefore be integrated into a self-powered device. This enables the use of hash functions in

ultra-low-power applications such as “Smart Dust” motes, RFIDs, and Piconet nodes. By virtue of the security and implementation features mentioned above, we believe that the proposed universal hash function together with the Toeplitz approach will fill an important gap in cryptographic hardware applications.

REFERENCES

[1] F. Bennett, D. Clarke, J. B. Evans, A. Hopper, A. Jones, and D. Leask, “Piconet: Embedded mobile networking,” *IEEE Personal Communications*, vol. 4, no. 5, pp. 8–15, Oct 1997.

[2] S. Sarma, D. Brock, and K. Ashton, “The networked physical world - proposals for engineering the next generation of computing, commerce & automatic identification,” MIT: Auto-ID Center,” White Paper, Oct 2000.

[3] J. Kahn, R. Katz, and K. Pister, “Next century challenges: mobile networking for ”smart dust”,” in *Proceedings of the fifth annual ACM/IEEE international conference on Mobile computing and networking.* ACM, 1999, pp. 271–278.

[4] S. Meininger, J. Mur-Miranda, R. Amirtharajah, A. Chandrakasan, and J. Lang, “Vibration-to-electric energy conversion,” *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 9, no. 1, pp. 64–76, Feb 2001.

[5] G. J. Simmons, Ed., *Contemporary Cryptology*. IEEE Press, 1992.

[6] W. Diffie and M. E. Hellman, “New Directions in Cryptography,” *IEEE Transactions on Information Theory*, vol. IT-22, pp. 644–654, 1976.

[7] J. L. Carter and M. Wegman, “Universal classes of hash functions,” *Journal of Computer and System Sciences*, vol. 18, pp. 143–154, 1978.

[8] ——, “New hash functions and their use in authentication and set equality,” *Journal of Computer and System Sciences*, vol. 22, pp. 265–279, 1981.

[9] J. Black, S. Halevi, H. Krawczyk, T. Krovetz, and P. Rogaway, “UMAC: Fast and secure message authentication,” in *Advances in Cryptology - CRYPTO ’99*, ser. Lecture Notes in Computer Science, vol. 1666. Springer-Verlag, 1999, pp. 216–233.

[10] P. Rogaway, “Bucket hashing and its application to fast message authetication,” in *Proceedings Crypto ’95*, ser. LNCS, D. Coppersmith, Ed., vol. 963. Springer-Verlag, 1995, pp. 29–42.

[11] D. W. Carman, P. S. Kruus, and B. J. Matt, “Constraints and approaches for distributed sensor network security,” NAI Labs, Secutity Research Division, Glenwood, MD, Technical Report, Sep 2000.

[12] M. Ramakrishna, E. Fu, and E. Bahcekapili, “A performance study of hashing functions for hardware applications,” in *Proceedings of the ICCT ’94 International Conference on Computing and Information*, 1994, pp. 1621–1636.

[13] H. Krawczyk, “LFSR-based hashing and authentication,” in *Advances in Cryptology - CRYPTO ’94*, ser. Lecture Notes in Computer Science, vol. 839. Springer-Verlag, 1994, pp. 129–139.

[14] V. Shoup, “On fast and provably secure message authentication based on universal hashing,” in *Advances in Cryptology - CRYPTO ’96*, ser. Lecture Notes in Computer Science, vol. 1109. New York: Springer-Verlag, 1996, pp. 74–85.

[15] S. Halevi and H. Krawczyk, “MMH: Software message authentication in the gbit/second rates,” in *4th Workshop on Fast Software Encryption*, ser. Lecture Notes in Computer Science, vol. 1267. Springer, 1997, pp. 172–189.

[16] P. Rogaway, “Bucket hashing and its applications to fast message authentication,” in *Advances in Cryptology - CRYPTO ’95*, ser. Lecture Notes in Computer Science, vol. 963. New York: Springer-Verlag, 1995, pp. 313–328.

[17] H. Krawczyk, “New hash functions for message authentication,” in *EUROCRYPT’95*, ser. Lecture Notes in Computer Science, vol. 921. Springer-Verlag, 1995, pp. 301–310.

[18] M. Etzel, S. Patel, and Z. Ramzan, “SQUARE HASH: Fast message authentication via optimized universal hash functions,” in *Advances in Cryptology - CRYPTO ’99*, ser. Lecture Notes in Computer Science, M. Wiener, Ed., vol. 1666. New York: Springer-Verlag, 1999, pp. 234–251.

[19] W. Nevelsteen and B. Preneel, “Software performance of universal hash functions,” in *EUROCRYPT’99*, ser. Lecture Notes in Computer Science, vol. 1592. Berlin: Springer-Verlag, 1999, pp. 24–41.

[20] R. J. McEliece, *Finite Fields for Computer Scientists and Engineers*, 2nd ed. Kluwer Academic Publishers, 1989.

[21] Y. Mansour, N. Nissan, and P. Tiwari, “The computational complexity of universal hashing,” in *22nd Annual ACM Symposium on Theory of Computing*. ACM Press, 1990, pp. 235–243.

[22] S. Devadas and S. Malik, “A survey of optimization techniques targeting low power VLSI circuits,” in *Proceedings of the 32nd ACM/IEEE Conference on Design Automation*, 1995, pp. 242–247.

[23] J. Rabaey and M. Pedram, *Low Power Design Methodologies*. Norwell, Massachusetts: Kluwer Academic Publishers, 1996.

[24] B. Parhami, *Computer Arithmetic: Algorithms and Hardware Designs*. Oxford University Press, 2000.

[25] K. M. Heal, M. L. Hansen, and K. M. Rickard, *Maple V Learning Guide*. New York: Springer Verlag, 1998.

[26] R. Amirtharajah and A. P. Chandrakasan, “Self-powered signal processing using vibration-based power generation,” *IEEE Journal of Solid-State Circuits*, vol. 33, no. 5, pp. 687–695, May 1998.

Jens-Peter Kaps received his Dipl.-Ing. in Electrical Engineering from the University of Applied Science, Munich, Germany in 1996 and his M.Sc. in Electrical and Computer Engineering (ECE) from Worcester Polytechnic Institute (WPI), Massachusetts in 1998. He worked for three and a half years as a senior engineer for GTE CyberTrust Inc., Needham, Massachusetts, before returning to WPI in order to pursue a Ph.D. degree in Electrical and Computer Engineering. His research interests include ultra-low-power cryptographic hardware design, computer arithmetic, efficient cryptographic algorithms as well as computer and network security. He is a student member of the IEEE Computer Society and the International Association of Cryptologic Research (IACR) professional societies as well as Eta Kappa Nu (HKN) International Honor Society for Electrical Engineers

Kaan Yüksel received the BSc degree in electrical and electronics engineering (EEE) from Middle East Technical University, Ankara, Turkey, in 2002 and the MSc in electrical and computer engineering (ECE) from Worcester Polytechnic Institute, Worcester, Massachusetts, in 2004. He is currently pursuing the PhD degree. His research interests include cryptography and information security with emphasis on message authentication codes, universal hash functions, extractors and hardware random number generators. He is a student member of the IEEE, Eta Kappa Nu (HKN) International Honor Society for Electrical Engineers, and the International Association of Cryptographic Research (IACR).

Berk Sunar received his BSc degree in Electrical and Electronics Engineering from Middle East Technical University in 1995 and his Ph.D. degree in Electrical and Computer Engineering (ECE) from Oregon State University in December 1998. After briefly working as a member of the research faculty at Oregon State University, Sunar has joined Worcester Polytechnic Institute as an Assistant Professor. He is currently heading the Cryptography and Information Security Laboratory (CRIS). Sunar received the National Science Foundation CAREER award in 2002. He organized the Cryptographic Hardware and Embedded Systems Conference (CHES) in 2004, and is the co-editor of CHES 2005. His research interests include finite fields, elliptic curve cryptography, low-power cryptography, and computer arithmetic. Sunar is a member of the IEEE Computer Society, the ACM, and the International Association of Cryptologic Research (IACR) professional societies.