A Systems Theoretic Approach to the Design of Scalable Cryptographic Hash Functions

Josef Scharinger

Johannes Kepler University, Institute of Computational Perception,
4040 Linz, Austria
Josef.Scharinger@jku.at

Abstract. Cryptographic hash functions are security primitives that compute check sums of messages in a strong manner and this way are of fundamental importance for ensuring integrity and authenticity in secure communications. However, recent developments in cryptanalysis indicate that conventional approaches to the design of cryptographic hash functions may have some shortcomings.
Therefore it is the intention of this contribution to propose a novel way how to design cryptographic hash functions. Our approach is based on the idea that the hash value of a message is computed as a message-dependent permutation generated by very special chaotic permutation systems, so called Kolomogorov systems. Following this systems theoretic approach we obtain arguably strong hash functions with the additional useful property of excellent scalability.

1 Introduction and Motivation

Cryptographic hash functions for producing checksums of messages are a core primitive in secure communication. They are used to ensure communication integrity and are also essential to signature schemes because in practice one does not sign an entire message, but the cryptographic checksum of the message.

All the cryptographic hash functions in practical use today (SHA-1, SHA-224, SHA-256, SHA-384 and SHA-512) are specified in the Secure Hash Standard (SHS, see [8]) and are based on ideas developed by *R. Rivest* for his MD5 message digest algorithm [9]. Unfortunately, recent attacks [14] on SHA-1 show that this design approach may have some shortcomings. This is the reason why the intention of this contribution is to deliver a radically different systems theory based approach to the design of scalable cryptographic hash functions.

The reminder of this contribution is organized as follows. In section 2 we explain the notion of a cryptographic hash function. Section 3 introduces the well-known class of continuous chaotic Kolomogorov systems, present a discrete version of Kolomogorov systems and analyze cryptographically relevant properties of these discrete Kolomogorov systems. Next, section 4 describes our novel approach to the design of cryptographic hash functions which is essentially based on the idea of computing a message check sum as a message dependent permutation generated by iterated applications of the discrete Kolmogorov systems

described in section 3. Finally, section 5 intends to justify the claim that our design of cryptographic hash functions based on systems theory constitutes a highly scalable approach to the development of cryptographic hash functions.

2 Cryptographic Hash Functions

2.1 The Concept of a Cryptographic Hash Function

Following [11], cryptographic hash functions come under many different names: one-way hash function, message digest function, cryptographic checksum function, message authentication code, and quite some more. Essentially a cryptographic hash function takes an input string and converts it to a fixed-size (usually smaller) output string.

In a more formal way, a cryptographic hash function $H(M)$ operates on an arbitrary-length plaintext (message) M and returns a fixed-length hash value $h = H(M)$, where h is of length N. While one can think of many functions that convert an arbitrary-length input and return an output of fixed length, a cryptographic hash function has to have additional characteristics:

- *one-way property*: given M, it is easy to compute h, but given h, it is hard to compute M
- *collision resistance*: given M, it is hard to find another message M', such that $H(M) = H(M')$ and even more it should be hard to find two arbitrary messages M_1 and M_2 such that $H(M_1) = H(M_2)$

It is perfectly obvious to see that any cryptographic hash function producing length N hash values can only offer order $\mathcal{O}(2^N)$ security with respect to fulfilling the one-way property. Even more, taking into consideration the so-called *birthday attack* [11], it follows that any cryptographic hash function can only offer order $\mathcal{O}(2^{N/2})$ security with respect to collision resistance. It is therefore essential to note that N defines an upper limit on security that is achievable by any length N cryptographic hash function. Accordingly it would be nice to have *scalable* hash functions where increasing N should be as simple as possible, a point we pay special attention to with our approach presented in this paper.

3 Chaotic Kolmogorov Systems

Among the most remarkable results of recent systems theory are novel findings on chaotic systems. There has been good progress in systems science concerning the analysis of complex dynamical systems and concepts like fractal dimension or strange attractors are now well understood. However, it is worth noting that the overwhelming majority of exiting systems is by definition of continuous type, so system states are in some power set of $\mathbb{R}$.

A fundamental property of chaotic systems is the fact that small deviations in inputs can completely alter the systems behavior. This immediately leads to

the problem that any approximations, as inherently involved by any digitization, may change systems behavior completely. Therefore, for practical digital applications of interesting chaotic systems it is essential to successfully bridge the gap from continuous type systems to discrete version that still preserve the essential properties present in the continuous case.

In our contribution we focus on the class of chaotic Kolmogorov systems [3, 6, 13]. This class has been of great interest to systems scientists for a long time due to some unique properties amongst which the outstanding degree of instability is particularly remarkable. It has been proven [2] that continuous Kolmogorov systems T_π guarantee ergodicity, exponential divergence and perfect mixing of the underlying state space for almost all valid choices of parameter π. Note that these properties perfectly match the properties of *confusion and diffusion* (as first defined by *C. Shannon* in [12]) that are so fundamental in cryptography.

3.1 Continuous Kolmogorov Systems

Continuous chaotic Kolmogorov systems act as permutation operators upon the unit square $\mathbb{E}$. Figure 1 is intended to give a notion of the dynamics associated with a specific Kolmogorov system parameterized by the partition $\pi = (\frac{1}{3}, \frac{1}{2}, \frac{1}{6})$. As can be seen, the unit square is first partitioned into three vertical strips according to $\frac{1}{3}, \frac{1}{2}, \frac{1}{6}$. These strips are then stretched to full width in the horizontal and squeezed by the same factor in the vertical direction and finally these transformed strips are stacked atop of each other. After just a few applications (see Fig. 1 from top left to bottom right depicting the initial and the transformed state space after 1, 2, 3, 6 and 9 applications of T_π) this iterated stretching, squeezing and folding achieves excellent mixing of the elements within the state space.

Fig. 1. Illustrating the chaotic and mixing dynamics associated when iterating a Kolmogorov system.

Formally this process of stretching, squeezing and folding is specified as follows. Given a partition $\pi = (p_1, p_2, \ldots, p_k)$, $0 < p_i < 1$ and $\sum_{i=1}^{k} p_i = 1$ of the unit interval $\mathbb{U}$ and stretching and squeezing factors defined by $q_i = \frac{1}{p_i}$. Furthermore, let F_i defined by $F_1 = 0$ and $F_i = F_{i-1} + p_{i-1}$ denote the left border of the vertical strip containing the point $(x, y) \in \mathbb{E}$ to transform. Then the continuous Kolmogorov system T_π will move $(x, y) \in [F_i, F_i + p_i) \times [0, 1)$ to the position

$$T_\pi(x, y) = (q_i(x - F_i), \frac{y}{q_i} + F_i). \tag{1}$$

It is well known and proven [2] that *for almost all valid choices of parameter* π the corresponding continuous Kolmogorov system T_π fulfills the following appealing properties:

- *ergodicity*: guarantees that almost any initial point approaches any point in state space arbitrarily close as the system evolves in time. Speaking in terms of cryptography this property can be considered as equivalent to *confusion* since initial (input) positions does not give any information on final (output) positions.
- *exponential divergence*: neighboring points diverge quickly at exponential rate in horizontal direction. Speaking in terms of cryptography this property can be considered as equivalent to *diffusion* since initially similar initial (input) positions rapidly lead to highly different final (output) positions.
- *mixing*: guarantees that all subspaces of the state space dissipate uniformly over the entire state space. Speaking in terms of cryptography this property can be considered as a perfect equivalent to *confusion and diffusion*.

Deducing from this analysis it can be concluded that continuous Kolmogorov systems offer all the properties desired for a perfect permutation operator in the *continuous* domain. Our task now is to develop a *discrete* version of Kolmogorov systems that preserves these outstanding properties. That is precisely what will be done in the next subsection.

3.2 Discrete Kolmogorov Systems

In our notation a specific discrete Kolmogorov system for permuting a data block of dimensions $n \times n$ shall be defined by a list $\delta = (n_1, n_2, \ldots, n_k)$, $0 < n_i < n$ and $\sum_{i=1}^{k} n_i = n$ of positive integers that adhere to the restriction that all $n_i \in \delta$ must partition the side length n.

Furthermore let the quantities q_i be defined by $q_i = \frac{n}{n_i}$ and let N_i specified by $N_1 = 0$ and $N_i = N_{i-1} + n_{i-1}$ denote the left border of the vertical strip that contains the point (x, y) to transform.

Then the discrete Kolmogorov system $T_{n,\delta}$ will move the point $(x, y) \in [N_i, N_i + n_i) \times [0, n)$ to the position

$$T_{n,\delta}(x, y) = (q_i(x - N_i) + (y \bmod q_i), (y \operatorname{div} q_i) + N_i). \tag{2}$$

As detailed in the preceding subsection, continuous Kolmogorov systems T_π are perfect (ergodic and mixing) permutation operators in the continuous domain. Provided that our definition of discrete Kolmogorov systems $T_{n,\delta}$ has the same desirable properties in the discrete domain, that would deliver a strong permutation operator inherently possessing the properties of confusion, diffusion and perfect statistics in the sense that permutations produced are statistically indistinguishable for truly random permutations. The analysis in the next subsection proofs exactly that this is true indeed.

3.3 Analysis of Discrete Kolmogorov Systems

As detailed in [10], the following theorem can be proven for discrete Kolmogorov systems T_{n,δ_r}:

Theorem 1. *Let the side-length $n = p^m$ be an integral power of a prime p. Then the application of discrete Kolmogorov systems T_{n,δ_r} leads to ergodicity, exponential divergence and mixing provided that at least $4m$ iterations are performed and lists δ_r used in every round r are chosen independently and at random. As an immediate consequence, this definitely is the case if at least $4\log_2 n$ rounds are iterated.*

For any cryptographic system it is always essential to know how many different keys are available to the cryptographic system. In our case of discrete Kolmogorov systems $T_{n,\delta}$ this reduces to the question, how many different lists $\delta = (n_1, n_2, \ldots, n_k)$ of n_i summing up to n do exist when all n_i have to part n?

As detailed in e.g. [1], a computationally feasible answer to this question can be found by a method based on formal power series expansion leading to a simple recursion relation. If $R = \{r_1, r_2, \ldots, r_m\}$ denotes the set of admissible divisors in ascending order, then c_n, the number of all lists δ constituting a valid key for $T_{n,\delta}$, is given by

$$c_n = \begin{cases} 0, & \text{if } n < r_1 \\ c_{n-r_1} + c_{n-r_2} + \ldots + c_{n-r_m} & \text{if } (n \geq r_1) \wedge (n \notin \{r_1, r_2, \ldots, r_m\}) \\ 1 + c_{n-r_1} + c_{n-r_2} + \ldots + c_{n-r_m} & \text{if } n \in \{r_1, r_2, \ldots, r_m\} \end{cases} \tag{3}$$

Some selected results are given in table 1. To fully appreciate these impressive numbers note that values given express the number of permissible keys for just one round and that the total number of particles in the universe is estimated to be in the range of about 2^{265}.

4 Hash Functions from Chaotic Kolmogorov Systems

Deducing from theorem 1, the following holds true:

- if random parameters δ_r are used and at least $4\log_2 n$ rounds are iterated, then any $n \times n$ square will be perfectly permuted by applying a sequence of transformations T_{n,δ_r}

n	c_n	n	c_n	n	c_n
4	1	8	5	16	55
32	5.271	64	47.350.055	128	$\approx 2^{50}$
256	$\approx 2^{103}$	512	$\approx 2^{209}$	1024	$\approx 2^{418}$

Table 1. Number of permissible parameters δ for parameterizing the discrete Kolmogorov system $T_{n,\delta}$ for some selected values of n

– this permutation is determined by the sequence of parameters δ_r

This immediately leads to the following idea how to calculate the hash value for a message M using discrete Kolmogorov systems T_{n,δ_r}:

– the bits of the message M can be interpreted as a sequence of parameters δ_r
– the application of a sequence of transforms T_{n,δ_r} will result in a permutation hash determined by message M

According to this principle, our algorithm for the calculation of a *Kolmogorov permutation hash* of length N for a message M works as described next.

4.1 Initialization

In the initialization phase all that has to be done is fill a square array of side length n (such that $n \times n = N$) with e.g. left half $N/2$ zeros and right half $N/2$ ones.

4.2 Message Schedule

Next we partition message M into blocks M_i (e.g. of size 512 bits). This is useful e.g. because this way the receiver of a message can begin calculation of hash values without having to wait for receipt of the entire message and additionally this keeps our approach in compliance with e.g. the HMAC algorithm [7] which demands an iterated hash function in its very definition.

Then we expand block M_i to get a corresponding expanded pseudo-random message block W_i. This can e.g. be done using linear congruence generators (LCGs, see [5]), linear feedback shift registers (LFSRs, see [4]) or the expansion mechanisms used in the Secure Hash Standard (SHS, see [8]) defined by NIST. All we demand is that this expansion has to deliver $\geq 4 \log_2 n$ M_i-dependent pseudo-random groups $g_{i,r}$ of bits interpretable as parameters $\delta_{i,r}$ (see following two subsections on interpretation and use of bit groups $g_{i,r}$).

4.3 Mapping Bit Groups onto Partitions

When the task is to map bit groups $g_{i,r}$ from M_i and W_i onto valid parameters $\delta_{i,r} = (n_1, n_2, \ldots, n_k)$ this can be accomplished in very simple ways. Just examine the following illustration:

M_i or W_i			
$g_{i,1}$	$g_{i,2}$	$g_{i,3}$	$\ldots$
0 1 . . . 0	1 1 . . . 0	0 1 . . . 1	. . . . .
$\delta_{i,1}$	$\delta_{i,2}$	$\delta_{i,3}$	$\ldots$

If one writes down explicitly the different partitions $\delta_{i,r}$ possible for various n (whose total number is given in Tab. 1) one immediately notices that the probability of a value n_i being contained in a partition decays exponentially with the magnitude of n_i. Therefore the following approach is perfectly justified. When iterating over bit groups $g_{i,r}$ and generating factors n_i for $\delta_{i,r}$, we interpret the smallest run of equal bits (length 1) as the smallest factor of n, namely $n_i = 2^1 = 2$, a run of two equal bits as factor $n_i = 2^2 = 4$, and so on.

There are two details to observe in the above outlined procedure of mapping bit groups $g_{i,r}$ onto valid partitions $\delta_{i,r}$:

- One point to observe in this procedure is that the sum of n_i's generated from bit groups $g_{i,r}$ this way has to equal n. Therefore one has to terminate a run as soon as an n_{j+1} would be generated such that $\sum_{i=1}^{j} n_i + n_{j+1} > n$. Then the maximum possible n_{j+1} (as a power of 2) still fulfilling the constraint has to be chosen, and the run length has to be reset to one.
- The other point to observe is that one iteration over $g_{i,r}$ may yield n_i summing up to less than n. In that case $g_{i,r}$ just has to be scanned iteratively until the n_i generated sum up to n indeed.

Observance of these two details will guarantee that valid parameters $\delta_{i,r}$ will always be generated for bit groups $g_{i,r}$.

4.4 Processing a Message Block

Processing message block M_i and corresponding expanded message block W_i is perfectly straightforward. Just iteratively take groups of bits $g_{i,r}$ first from M_i then from W_i, map them onto parameters $\delta_{i,r}$, permute square array according to $T_{n,\delta_{i,r}}$, and finally rotate the array by $g_{i,r}$ mod N to avoid problems with fixed points $(0,0)$ and $(n-1,n-1)$. All one has to care about in this simple scheme is that groups $g_{i,r}$ taken from M_i must have sizes k, such that 2^k is lower or equal to the number of permissible keys (see Tab. 1) for $T_{n,\delta_{i,r}}$ to avoid collisions, and that groups $g_{i,r}$ taken from W_i must have sizes k, such that 2^k is greater or equal to the number of permissible keys for $T_{n,\delta_{i,r}}$ to ensure perfect mixing according to theorem 1.

Applying this procedure for all message blocks M_i of message M will result in excellent chaotic mixing of the square array *in strong dependence on message* M.

4.5 Reading Out the Message Digest

Finally, reading out the state of array reached after processing all M_i yields a strong checksum of length $N = n \times n$ for message M.

5 Scalability

Some readers might wonder why our description of Kolmogorov permutation hashes as specified in section 4 does not fix a specific value N for the length of hash values produced by our approach. The reason is simple: we want our approach to the design of cryptographic hash functions to be as generic as possible. As already indicated in the title of this contribution, we are aiming at the development of *scalable* cryptographic hash functions.

To understand why this scalability is so important, recall from section 2 that it is a fact that an N bit hash function can only offer security up to level $\mathcal{O}(2^{N/2})$ [11]. Consequently, as computing power is increasing steadily, it may become desirable to increase the length of hash values produced without having to redesign the hash function.

In our scheme, increasing the length and thus achieving remarkable scalability is straightforward. By just changing the size of the underlying square array from $n \times n$ to $2n \times 2n$, the length of hash values produced is increased by 4. Obviously, this involves minor modifications to block expansion and bit group partitioning as explained and specified in section 4, but besides these small changes, the same algorithm can be used.

References

1. M. Aigner. *Kombinatorik*. Springer Verlag, 1975.
2. V.I. Arnold and A. Avez. *Ergodic Problems of Classical Mechanics*. W.A. Benjamin, New York, 1968.
3. S. Goldstein, B. Misra, and M. Courbage. On intrinsic randomness of dynamical systems. *Journal of Statistical Physics*, 25(1):111–126, 1981.
4. Solomon W. Golomb. *Shift Register Sequences*. Aegan Park Pr., 1981.
5. Donald E. Knuth. *The Art of Computer Programming*. Addison-Wesley, 1998.
6. Jürgen Moser. *Stable and Random Motions in Dynamical Systems*. Princeton University Press, Princeton, 1973.
7. NIST. Keyed-Hash Message Authentication Code (HMAC). FIPS 198, March 2002.
8. NIST. Secure hash standard (SHS). FIPS 180-2, August 2002.
9. R.L. Rivest. The MD5 message digest function. RFC 1321, 1992.
10. Josef Scharinger. An excellent permutation operator for cryptographic applications. In *Computer Aided Systems Theory – EUROCAST 2005*, pages 317–326. Springer Lecture Notes in Computer Science, Volume 3643, 2005.
11. Bruce Schneier. *Applied Cryptography*. Addison-Wesley, 1996.
12. C.E. Shannon. Communication theory of secure systems. *Bell System Technical Journal*, 28(4):656–715, 1949.
13. Paul Shields. *The Theory of Bernoulli Shifts*. The University of Chicago Press, Chicago, 1973.
14. Xiaoyun Wang, Yiqun Lisa Yin, and Hongbo Yu. Finding collisions in the full SHA-1. In *CRYPTO*, 2005.