About a new generation of block ciphers and hash functions - DN and HDN

Vlastimil Klíma[*]

v.klima@volny.cz

Independent consultant
http://cryptography.hyperlink.cz
Prague, Czech Republic

Abstract

Antoine Joux, who discovered in 2004 the hash functions generic problem (multi-collisions), said at the SECOND CRYPTOGRAPHIC HASH WORKSHOP, USA, August 24 - 25, 2006: "We do not understand what we are doing and we do not really know what we want".

The [16] and [17] shows for the first time that to build a hash function from the classical block cipher is a vain effort like squaring the circle. This is also the real reason for the current problems of hash functions.

In [16] and [18] we have designed something weird – a symmetric block cipher, whose encryption key can be revealed to an attacker and we gave to this new cryptographic primitive a little bit confusing name: Special Block Cipher.

In 1975, a similar idea in another context triggered a revolution in cryptography and gave rise to a new branch: Public Key Cryptography. At that time encryption functions were created where an attacker could know the encryption key, which had until then seemed foolish.

Special block ciphers have much stricter requirements: an attacker can select and discretionarily tamper with the key, which seems even more foolish.

The special block cipher was designed as a new cryptographic primitive and by means of this primitive also a new family of hash functions SNMAC. SNMAC functions have publicly known design criteria and approach a random oracle in the limit. They are computationally resistant against pre-image and collision attacks and different special block cipher instances can be used in their design.

In this paper, we present the very first special block cipher family: the Double Net DN(n, k)-ρ with n-bit block, k-bit key and ρ rounds. And, based on DN, we define hash functions family HDN(n, k)-ρ with n-bit hash code which hashes messages per k–n bits blocks.

We introduce and propose to use DN(512, 8192)-10 and HDN(512, 8192)-10 as example instances. These functions are ready-to-use with speeds only 2-3 times lower than SHA-512 and Whirlpool.

Keywords: special block cipher, hash function.

1 Introduction

The classical block cipher is a cryptographic primitive designed to protect the plaintext and its structure in the ciphertext using the secret encryption key. The fact that an attacker does not know the secret key is essential for high-speed encryption in the classical block cipher design. This is the first important fact.

So-called "*Preparation of Key*" phase (the key expansion procedure) is very simple for most classical block ciphers. For instance, DES uses a simple "*copy*" function. While AES employs a weak non-linear transformation. The majority of block ciphers use weak non-linear or simple functions. This is the second significant fact.

[*] This paper presents some parts of the projects ST20052006018 and ST20052005017 for Czech NSA.

It was crucially exploited in the attacks on MD and SHA hash function families. The weak non-linear functions allowed, in many places, to control precisely the inner state of the hash function by a pre-defined strategy (differential path). Strong non-linear functions would not have allowed it. Because a compression function does not use any secret element, an attacker knows all inputs of the underlying classical block cipher and can tamper with them. Consequently they could even tamper with the encryption key. This is the third important fact.

All these three facts become weaknesses when a classical block cipher, designed to encrypt, is used in hash function as a one-way function.

We presume that the reason for the current problems with hash functions is the usage of classical block ciphers (originally designed for totally different purposes) as a compression function. A classical block cipher is a totally different primitive to the one-way function as we can see in Table 1. Therefore a hash function can not efficiently be based on it and it is necessary to use another cryptographic primitive.

In [16], [17] and [18] we have designed for this purpose a new cryptographic primitive (confusingly called special block cipher) and on its base a hash function of the SNMAC family.

In this paper, we present the first family of the special block ciphers DN and class of hash functions HDN based on them.

Classical Block Cipher	**Compression Function**
contains an element unknown to an attacker	an attacker knows all inputs and is able to spool them
is meant to hide the plaintext structure and content in the ciphertext, based on a secret element (unknown to an attacker)	is meant to hide all structure and content of all inputs in the output, based on a public function
is a permutation for fixed-key	is a random transformation
it is invertible	one-wayness is needed
it is easy to create collisions	collision resistance is needed

Table 1: Main differences between classical block ciphers and compression functions.

2 Functions family DN(n, k)-ρ description

DN(n, k)-ρ is n-bit block cipher with k-bit encryption key K and ρ (big) rounds, where ρ† is a security parameter.

DN consists of two functions, the key expansion Φ and the product cipher Π. The basic idea behind the DN double net is that the keys *a, b, …, z* for the sub-ciphers of the product cipher $\Pi = B_z \bullet \ldots \bullet B_b \bullet B_a$ are generated by strong block cipher Φ. With increasing number of rounds, the keys (*a*, *b*,…) and (…, *y*, *z*) become computationally indistinguishable from independent random variables, since they are in plaintext-ciphertext relation for the block cipher Φ. Thus, the block ciphers (B_a, B_b , ...) and (... B_y, B_z) themselves become computationally indistinguishable from (independent) random block ciphers. As the function Φ is a strong block cipher only on columns of key array RK (see Fig. 1), reasonable efficiency is achieved. The function Π mixes the columns of array RK with the plaintext.

† The variable ρ is denoted as rho in the source code [Kl07]

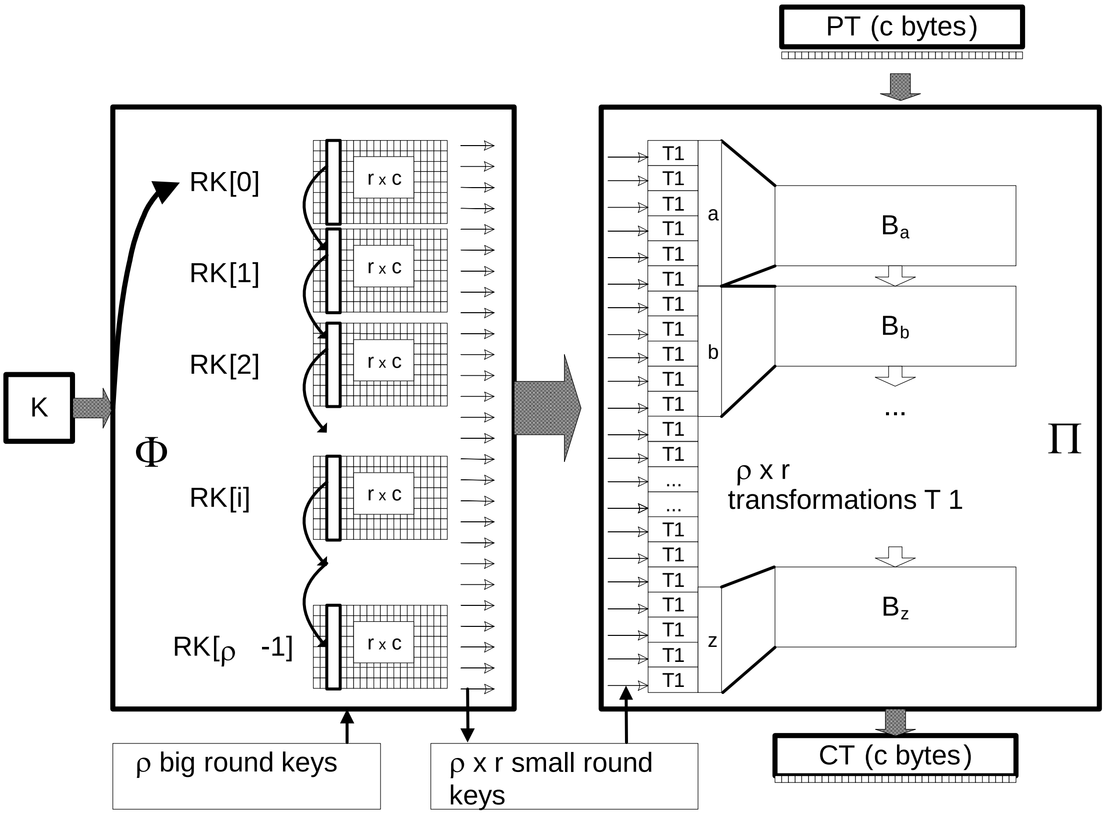

Figure1: Special block cipher family DN.

2.1 Function Φ

The scheme is described on byte level. The number of the bytes in the plaintext is denoted as $c = n/8$. The function Φ works with three-dimensional ρ x r x c array of bytes RK[i][j][t], i = 0, ..., ρ - 1, j = 0, ..., r - 1, t = 0, ..., c - 1 which is called round keys array. The first index (i) determines the big round key RK[i] as two-dimensional r x c array. The big round key RK[i] consists of r small round keys RK[i][j], j = 0, ..., r - 1. Small round key RK[i][j] is one row of the big round key and has c bytes RK[i][j][t], t = 0, ..., c - 1. The key K is the input to the function Φ. It is written into the first big round key RK[0] (left to right and up to down). From the first big round key, the function Φ progressively generates remaining ρ - 1 big round keys RK[i], i = 1, ..., ρ - 1 using so called column transformations, see Fig. 2.

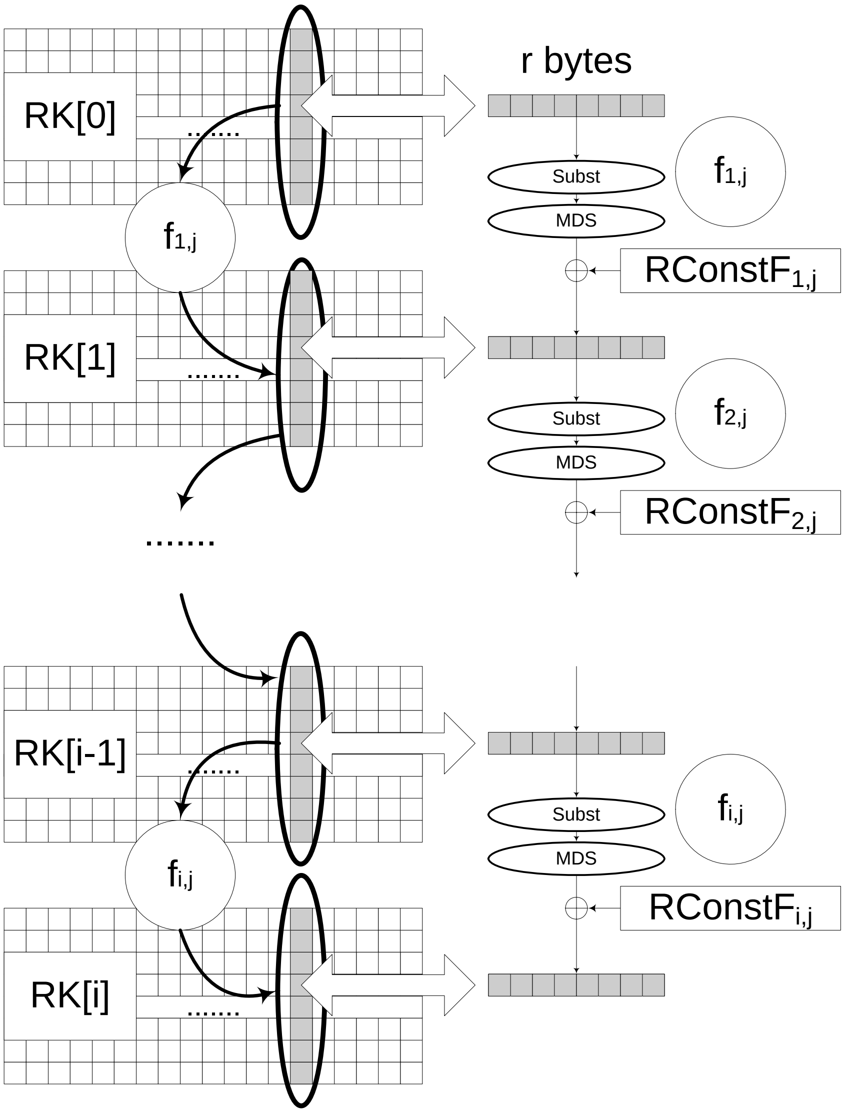

Figure 2: Column transformation.

2.2 Function Π

The function Π is a product of ρ x r elementary transformations T1, $\Pi = \Pi_{i\,=\,\rho\,-\,1,\,...,\,0}\ \Pi_{j\,=\,r\,-\,1,\,...,\,0}\ \mathrm{T1}_{i,j}$, where $\mathrm{T1}_{i,j}$ uses small round key RK[i][j], i = 0, ..., ρ - 1, j = 0, ..., r - 1. The output from one transformation T1 is the input to another transformation T1. Input to the function Π is the input to the first transformation T1, the output from the last transformation T1 is the output from the function Π.

2.2.1 Transformation T1

Each transformation $\mathrm{T1}_{i,j}$, i = 0, ..., ρ - 1, j = 0, ..., r - 1, consists of a substitution and a permutation on byte level, a linear transformation on bit level (not convertible to byte level) and small round key and round constant additions. All these variables can be different for different transformations $\mathrm{T1}_{i,j}$.

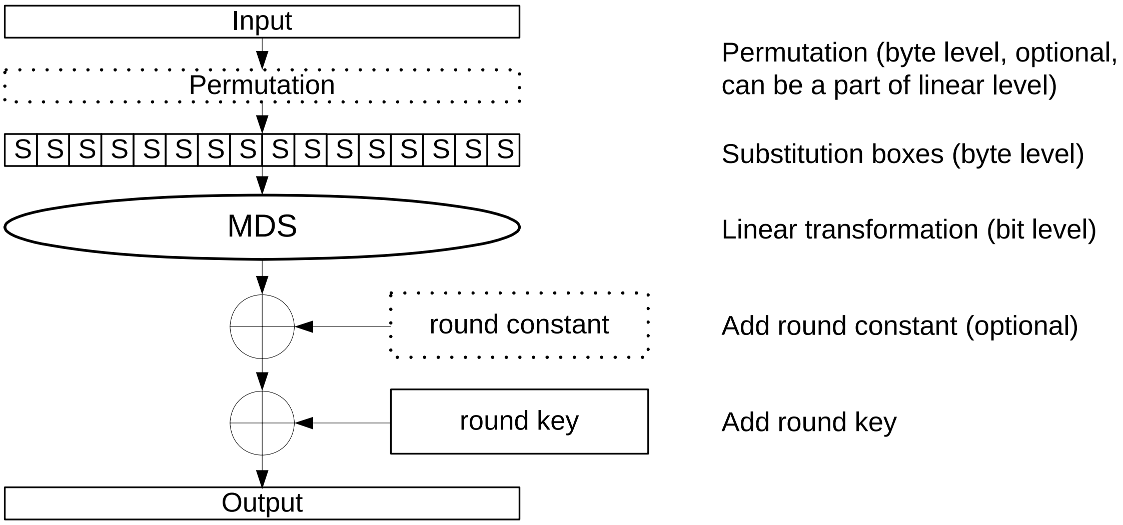
Input
Permutation
S S S S S S S S S S S S S S S S
MDS
round constant
round key
Output
Permutation (byte level, optional, can be a part of linear level)
Substitution boxes (byte level)
Linear transformation (bit level)
Add round constant (optional)
Add round key

Figure 3: Transformation T1.

2.3 Variable parameters of DN block cipher family DN

DN is a general scheme based on two SP networks Φ and Π. DN(*n*, *k*)-*ρ* has variable all following parameters:

Main dimensions:

- *n*, (plaintext length in bits; length of the hash code),
- *k*, (key *K* length in bits),
- *ρ*, number of big rounds,

Function Φ:

- S-boxes (mapping a byte on a byte), matrices MDS, round constants
- (optional) final key permutation KeyPerm in the key array RK,

Function Π:

- S-boxes (mapping a byte on a byte), matrices MDS, round constants
- all permutations in transformations T1.

All parameters and building blocks can be chosen different and with high freedom. However, there are some rules that the building blocks have to respect ([18]), briefly:

- function Π is a strong block cipher,
- all column transformations of Φ are strong block ciphers (with a fixed key !), they are pair wise different if possible,
- functions Φ and Π do not share any S-box, all of the S-boxes have good linear and differential characteristics and they are generated non-algebraically, (pseudo)randomly if possible,
- matrices used by the functions Φ and Π are all MDS type matrices (maximum distance separable).

3 Network Π construction

3.1 Π as product of block ciphers B

The function Π is the product of the block ciphers B, each employing several T1 rounds (several small round keys), i.e. $\Pi = B_z \bullet B_y \bullet \dots \bullet B_b \bullet B_a$.

3.2 Network Π S-boxes

Denote p_B (q_B) as the maximum value of the maximal differential probability (maximal linear probability, respectively) taken over all S-boxes used in the function B. Smaller the values of p_B and q_B are, more resistant against linear and differential cryptanalysis the function B becomes, thus less rounds is sufficient.

3.3 Network Π resistance against DC and LC

As B can be seen as big box $B: \{0, 1\}^n \rightarrow \{0, 1\}^n$, $n = 8c$, its resistance against differential (DC) and linear (LC) cryptanalysis is estimated by its maximum differential and maximum linear probabilities estimates, according to the following theorem.

Theorem 1. Block cipher B resistance against DC and LC.

If B is constructed as nested SP network according to [18], the following holds

$DP^B \le (p_B)^c$,

$LP^B \le (q_B)^c$.

Proof. [18].

4 Network Φ construction

4.1 S-boxes SubsF$_{i,j,t}$

Let's denote p_Φ (q_Φ) as the maximum value DP^S (LP^S) over all S-boxes SubsF$_{i,j,t}$ ($i = 1, \dots, \rho - 1$, $t = 0, \dots, c - 1$, $j = 0, \dots, r - 1$) used in the function Φ. Smaller these values are, less big rounds ρ DN may have. Random or pseudo-random S-boxes with sufficient resistance against linear and differential cryptanalysis are the ideal choice.

4.2 Network Π resistance against DC and LC

Theorem 2. Block cipher F$_t$, $t = 0, \dots, c - 1$, resistance against DC and LC.

Joining of two consecutive rounds of block cipher $F_t = f_{\rho-1,t} \bullet \dots \bullet f_{2,t} \bullet f_{1,t}$ (according to [18]) creates an SDS network with

$DP^{SDS} \le (p_\Phi)^r$,

$LP^{SDS} \le (q_\Phi)^r$.

Proof. [18].

Note. The final key permutation KeyPerm is also variable in the function Φ. From the security point of view, its usage isn't necessary; its goal is to improve diffusion of round keys in the function Π [18].

5 Double Net as a strengthened encryption algorithm

In the case of classical block ciphers, in the beginning it was assumed the attacker has no knowledge about the plaintext, later it was admitted he could know or even choose some of its parts. Currently, full control over the plaintext and ciphertext is taken into account. As an answer to these possibilities of the attacker, strong non-linear functions processing the plaintext were introduced.

Unfortunately, it was and still is assumed the attacker does not know the encryption key and has no means to manipulate with it. The technology development and the birth of various encryption devices (smart-cards, SSL servers, cryptographic modules, libraries, etc.) provide attacker with new possibilities weakening both of these original assumptions – not knowing the key and the impossibility to manipulate with it, as well. Side channel attacks are good example of these possibilities.

The progress in the decades to come will undoubtedly show progress in the key exploiting attacks. As an answer to these possibilities of the attacker, strong non-linear functions processing the key should be introduced also.

Special block cipher is a strong shield against these attacks. So, in this context, it is possible to use it also for encryption.

The key in DN algorithm used for the encryption will not usually be as long as the key in DN algorithm used for hashing. In the case of encryption the array *r* x *c* can be relatively small and the dimension *c* (plaintext width in bytes) can be relatively small, as well. A typical 128-bit block cipher with 256-bit key, i.e. *c* = 16 and *r* = 2 can be used as an example. The column transformation principles can be preserved even when several neighbouring columns are joined and understood as one "thicker column" (2*r* bytes). The column transformation F is then applied on this "thicker column".

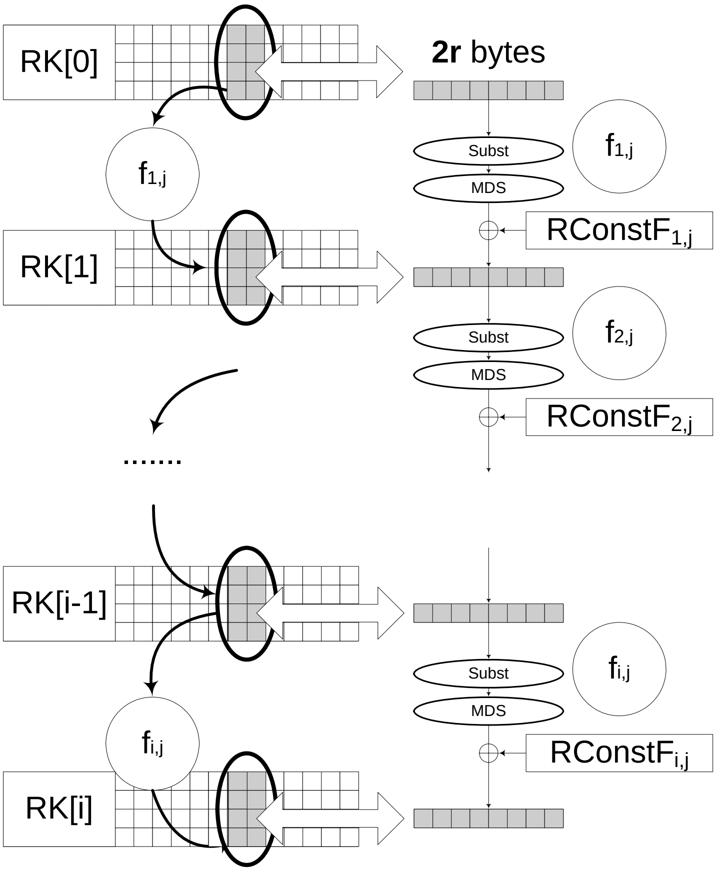

Figure 4: Column transformation principle applied on several columns.

6 Number of rounds in DN and hashing speed

The quality of substitution boxes and the dimensions of the round keys used in the function Φ determine the relationship between the number of rounds and the estimate for the resistance of Φ against DC and LC. Using current S-boxes from the algorithm Whirlpool [1] we set the number of rounds to 10 for the function DN(512, 8192). If higher quality S-boxes is used, the number of rounds can be lowered to as few as 6. However, we can say HDN(512, 8192)-10 is roughly 3 times slower than SHA-512 (and Whirlpool) and HDN(512, 8192)-6 roughly 2 times slower than SHA-512.

7 Description of hash function HDN(512, 8192)

If DN(512, 8192) is used in a hash function following the construction SNMAC [16], hash function HDN(512, 8192) is obtained with 512-bit code, processing the blocks of 7680 bits.

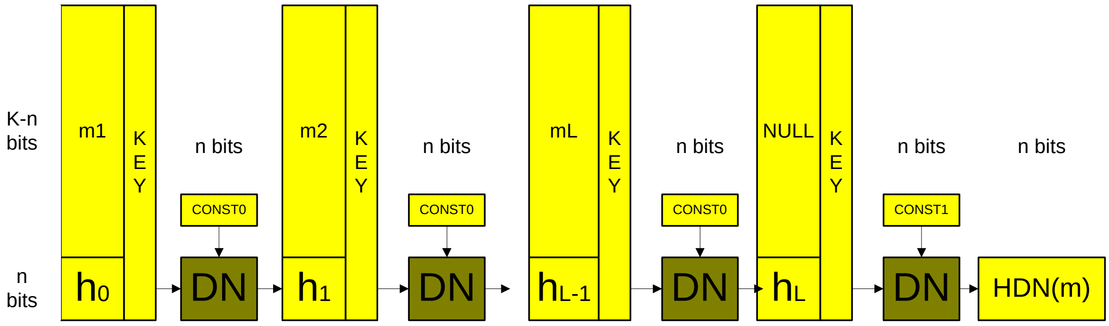

Figure 5: HDN(512, 8192) defined as SNMAC based on special block cipher DN(512, 8192)

Definition. Hash function HDN(512, 8192) is a SNMAC type hash function [16] based on special block cipher DN(512, 8192). It has n-bit hash code (n = 512), k bit key (k = 8192) and processes k - n bit data blocks (k - n = 7680). It employs compression function f and final modification function g, where

f: $\{0, 1\}^k \rightarrow \{0, 1\}^n : X \rightarrow E_X(\mathrm{Const}_0)$,

g: $\{0, 1\}^n \rightarrow \{0, 1\}^n : X \rightarrow E_{X \,||\, \mathrm{NULL}}(\mathrm{Const}_1)$,

and E is DN(512, 8192).

Const_0 and Const_1 are different constants and NULL is an array of k - n zero bits.

Message hashing is completed in three steps.

Step 1. Padding

Message m being hashed is padded by this (bit) string: a single bit 1, the least possible amount of bits 0 and 128 bit long number D (expression the binary length of m), so that the final message length could be expressed as $L(k - n)$ bits, for L an integer. The bit and byte orientation is the same as in SHA-512 standard, i.e. the last bit of block m_L contains the least signification bit of number D. The padded message is divided into L blocks of k - n bits, $m = m_1 \,||\, \ldots \,||\, m_{L-1} \,||\, m_L$. The same padding is used in function SHA-512.

Step 2. Iteration

$h_i = f(h_{i-1} \,||\, m_i)$, $i = 1, \ldots, L$,

where h_0 is constant initialization value (IV).

Step 3. Final modification

$\mathrm{SNMAC}(m) = \mathrm{g}(h_L)$.

8 Conclusion

We showed that employing a classical block cipher in hash functions design is the real reason for the current problems of all hash functions.

In [16] and [18] we have designed new cryptographic primitive - Special Block Cipher. It is a symmetric block cipher, whose encryption key can be revealed to an attacker. Moreover, an attacker can select and discretionarily tamper with the key.

Using special block cipher we proposed a new family of hash functions of the type SNMAC ([16], [17]). The design criteria of SNMAC hash functions are publicly known. Limitly, these functions approach a random oracle, they are computationally resistant against pre-image and collision attacks, and different special block cipher instances can be used in their design.

In this paper, we present the first special block cipher family DN and the first family of hash functions HDN following the SNMAC concept. It turns out these are not just theoretical concepts, but practically employable functions with speeds only 2-3 times lower than SHA-512 and Whirlpool.

Basic idea behind the special block cipher DN is simple – contrary to classical block cipher approach, the same attention is paid to key and plaintext processing.

Once the special block cipher concept is examined and accepted in hash functions, it can be used in advance in its original purpose – data encryption. The employment of these stronger functions might not seem as a must in the present, but it probably will be in the future. In the hash functions, it is a necessity today already.

Acknowledgement. The author thanks Tomáš Rosa, Martin Hlaváč and Alexandr Šnajdar for many useful comments to the drafts of this paper and for translating this paper from Czech.

Note. DN and HDN source codes are available on the homepages
http://cryptography.hyperlink.cz/SNMAC/SNMAC_EN.html,
http://cryptography.hyperlink.cz/SNMAC/SNMAC_CZ.html.

References

[1] P. Barreto, V. Rijmen, The Whirlpool Hashing Function, (Revised on May 24, 2003), ISO norm ISO/IEC 10118-3 (changed S-box) http://planeta.terra.com.br/informatica/paulobarreto/whirlpool.zip and the original specification from September 2000 (original S-box), https://www.cosic.esat.kuleuven.be/nessie/workshop/submissions/whirlpool.zip

[2] E. Biham, New Types of Cryptanalytic Attacks Using Related Keys, EUROCRYPT 1993, pp. 398-409, LNCS 765, Springer-Verlag, 1993.

[3] E. Biham, O. Dunkelman, N. Keller, Rectangle Attacks on 49-Round SHACAL-1, FSE 2003, pp. 22 - 35, LNCS 2887, Springer-Verlag, 2003.

[4] E. Biham, On Matsui's Linear Cryptanalysis, EUROCRYPT'94, LNCS 950, pp. 341-355, Springer-Verlag, 1995.

[5] E. Biham, A. Shamir, Differential Cryptanalysis of DES-like Cryptosystem, Journal of Cryptology, Vol.4, pp. 3-72, 1991.

[6] E. Biham, A. Shamir, Differential Cryptanalysis of Snefru, Khafre, REDOC-II, LOKI and Lucifer, CRYPTO'91, LNCS 576, pp. 156-171, Springer-Verlag, 1992.

[7] E. Biham, O. Dunkelman, N. Keller, Related-Key Boomerang and Rectangle Attacks, EUROCRYPT 2005, LNCS 3494, pp. 507–525, 2005.

[8] E. Biham, O. Dunkelman, N. Keller, Related-Key Impossible Differential Attacks on 8-Round

AES-192, CT-RSA 2006, LNCS 3860, pp. 21–33, Springer-Verlag, 2006.

[9] J. Daemen, Cipher and hash function design strategies based on linear and differential cryptanalysis, Doctoral Dissertation, March 1995, K.U. Leuven.

[10] S. Hong, S. Lee, J. Lim, J. Sung, D. Cheon, I. Cho, Provable Security against Differential and Linear Cryptanalysis for the SPN Structure, FSE 2000, LNCS 1978, pp. 273 - 283, Springer-Verlag, 2000.

[11] S. Hong, J. Kim, S. Lee, B. Preneel, Related-Key Rectangle Attacks on Reduced Versions of SHACAL-1 and AES-192, FSE 2005, LNCS 3557, pp. 368–383, Springer-Verlag, 2005.

[12] K. Chun, S. Kim, S. Lee, S.H. Sung, S. Yoon, Differential and linear cryptanalysis for 2-round SPNs, Information Processing Letters, Vol. 87 (2003), pp. 277 - 282.

[13] J. Kang, S. Hong, S. Lee, O. Yi, Ch. Park, J. Lim, Practical and provable security against differential and linear cryptanalysis for substitution-permutation networks, ETRI Journal, 23(4):158–167, 2001.

[14] J. Kim, G. Kim, S. Hong, S. Lee, D. Hong, The Related-Key Rectangle Attack-Application to SHACAL-1, ICISP 2004, LNCS 3108, pp. 123-136, Springer - Verlag, 2004.

[15] J. Kim, A. Biryukov, B. Preneel, S. Lee, On the Security of Encryption Modes of MD4, MD5 and HAVAL, Cryptology ePrint Archive: Report 2005/327, September - October 2005, ICICS 2005, LNCS 3783, Springer-Verlag, http://eprint.iacr.org/2005/327.pdf.

[16] V. Klima, A New Concept of Hash Functions SNMAC Using a Special Block Cipher and NMAC/HMAC Constructions, IACR ePrint archive Report 2006/376, October, 2006, http://eprint.iacr.org/2006/376.pdf

[17] V. Klima, New generation of hash functions SNMAC, Santa´s Crypto Get-Together, MKB 2006, Prague, December 2006, presentation on http://cryptography.hyperlink.cz/2006/MKB_2006_snmac.ppt, paper on http://cryptography.hyperlink.cz/2006/Klima_mkb_2006.pdf.

[18] V. Klima, Special block cipher family DN and new generation SNMAC-type hash function family HDN, homepage http://cryptography.hyperlink.cz/SNMAC/SNMAC_EN.html, IACR ePrint archive: Report 2007/050, February, 2007, http://eprint.iacr.org/2007/050.pdf.

[19] X. Lai, J. Massey, S. Murphy, Markov Ciphers and Differential Cryptanalysis, EUROCRYPT'91, LNCS 547, pp 17-38, Springer-Verlag, 1992.

[20] M. Matsui, Linear cryptanalysis method for DES cipher, EUROCRYPT' 93, LNCS 765, pp. 386-397, Springer-Verlag, 1993.

[21] M. Matsui, The first Experimental cryptanalysis of DES, CRYPTO'94, LNCS 839, pp. 1-11, Springer-Verlag, 1994.

[22] K. Nyberg, L. Knudsen, Provable security against a differential attack, CRYPTO'92, LNCS 740, pp. 566-574, Springer-Verlag, 1992.

[23] K. Nyberg, Linear Approximation of block ciphers, EUROCRYPT'94, LNCS 950, pp. 439-444, Springer-Verlag, 1994.

[24] J. Plank, Y. Ding, Note: Correction to the 1997 tutorial on Reed-Solomon coding, Software: Practice and Experience, Volume 35, Issue 2, pp. 189-194, 2005, http://www.cs.utk.edu/~plank/plank/papers/SPE-9-97.html.

[25] V. Rijmen, J.Daemen et al, The cipher SHARK, FSE´97, LNCS 1267, pp. 137-151, Springer-Verlag, 1997.

[26] R. Roth, Introduction to Coding Theory, Cambridge University Press, 2006, p. 148.

[27] F. Sano, K. Ohkuma, H. Shimizu, S. Kawamura, On the security of nested SPN cipher against the differential and linear cryptanalysis, IEICE Trans. Fundamentals, Vol. E86-A, No.1, January 2003, pp. 37 - 46.