S-FSB: An Improved Variant of the FSB Hash Family

Mohammed Meziani, Özgür Dagdelen, Pierre-Louis Cayrel, and Sidi Mohamed El Yousfi Alaoui

CASED – Center for Advanced Security Research Darmstadt,
Mornewegstrasse 32, 64293 Darmstadt, Germany
{mohammed.meziani,oezguer.dagdelen,pierre-louis.cayrel,elyousfi}@cased.de

Abstract. In 2003, Augot et al. introduced the Fast Syndrome-Based hash family (in short FSB), which follows the generic construction of Merkle-Damgård and is based on the syndrome decoding problem. In 2007, Finiasz et al. proposed an improved version of FSB. In this work, we propose a new efficient hash function, which incorporates the ideas of FSB and the sponge construction introduced by Bertoni et al. Our proposal is up to 30 % faster in practice than FSB. Its security is related on the Regular Syndrome (RSD) Decoding problem, which is proven NP-complete.

Keywords. cryptographic hash functions, provable security, syndrome decoding.

1 Introduction

A hash function maps strings of any length into short strings of fixed length, called *hash* or *digest*. For practical uses, hash functions should be easy to compute. That is, computing the hash value of a message m should be feasible in time polynomial in the size of m. Furthermore, if the hash function additionally satisfies certain security properties, it becomes a powerful tool for cryptographic applications such as digital signatures, conventional message authentication, password protection and pseudo-random number generation.

Over the last years, a long list of hash functions has been proposed in the literature. Following cryptanalytical advances, most of the widely used in practice (e.g. SHA-1) have been found to be insecure [12,26]. This has called into question the long-term security of later algorithms that share a similar design like SHA-2 family [23]. As a reaction, in 2007 the US National Institute of Standards and Technology (NIST) has opened a public competition, called SHA-3 (or the Advanced Hash Standard (AHS)), to develop new families of hash functions. Initially, 64 candidates have been submitted following different design principles, and only 4 of the competing designs passed to the third (and final) round of the contest. One of the first round submissions is the Fast Syndrome-Based hash Function (FSB) introduced first by Augot, Finiasz, and Sendrier [1] in 2003 and improved by Finiasz, Gaborit, and Sendrier [16] in 2007. The FSB is still unbroken up to now. It has a security reduction to NP-complete problems from coding theory, that are believed to be difficult on average. However, the main drawback of FSB is the efficiency issue because it is slower than other competing hash functions. For that reason, FSB was removed since the second round. We compensate for this disadvantage and speed up the process of hash computing by following a different design principle.
Instead of building the hash function upon the Merkle-Damgård design principle [22,15], we improve FSB further following the sponge construction, which is used in many hash functions like the SHA-3-finalist Keccack hash function [9].

Our contribution. In this paper we describe a new code-based hash function following the sponge construction. The mapping within FSB applied in a sponge hash function scheme results in a more efficient hash family than the origin FSB. We come up with detailed security analysis, including collision resistance and (second) preimage resistance.
Even though we reuse the transformation within FSB for our hash family, we show that their security results cannot be conveyed directly. Furthermore, the security analysis for a hash function built upon a sponge construction are given only for assuming random transformation. We go into detail how the security reacts when instantiated with a transformation given in FSB. Upon this analysis and the current state of the art

in cryptanalytic algorithms tackling the hard underlying problems within our scheme, we propose some parameters for fast and secure hashing.

Organization. The paper is organized as follows. Section 2 provides preliminaries. In Section 3 we briefly describe the related works that help us to construct our proposal. In Section 4 we present our construction. We analyze its security in Section 5. In Section 6 we propose some set of parameters for fast hashing. Section 7 presents the performance evaluation results of our proposal.

2 Preliminaries

Properties of Hash Functions. Hash functions are functions that map bit strings of arbitrary length into short fixed bit strings of length l. Besides the compressing property, "good" hash functions fulfill further properties like collision resistance and onewayness. Informally, collision resistance states that it is infeasible to find two distinct input values mapping to the same string when applied on the hash function. Furthermore, onewayeness states that it is infeasible to return a pre-image when given an output of the hash function (hash value).
Next, we state formal definitions of collision resistance and (second) pre-image resistance.

Definition 1 (Security Properties for Hash Functions). *A family of hash functions* $H_k : \{0,1\}^* \to \{0,1\}^l$ *with* $k \in \mathcal{K}$ *for some finite set* $\mathcal{K}$ *is* collision resistant *if for any probabilistic polynomial-time (PPT) algorithm* $\mathcal{A}$ *the probability that the experiment* $\mathsf{Col}^H_{\mathcal{A}}$ *evaluates to* 1 *is negligible (as a function of* λ*).* H_k *is* pre-image (resp. second pre-image) resistant *if for any PPT algorithm* $\mathcal{A}$ *the probability that the experiment* $\mathsf{PreImg}^H_{\mathcal{A}}$ *(resp.* $\mathsf{SecPre}^H_{\mathcal{A}}$*) evaluates to* 1 *is negligible (as a function of* λ*). Let* $\mathsf{Kg} : \lambda \mapsto k$ *be a generator for the keys of hash function* H_k *where* $k \in \mathcal{K}$*.*

Experiment $\mathsf{Col}^H_{\mathcal{A}}(\lambda)$
$k \leftarrow \mathsf{Kg}(\lambda)$
$(x, x') \leftarrow \mathcal{A}(k)$
Return 1 *iff* $x \neq x'$
and $H_k(x) = H_k(x')$.

Experiment $\mathsf{PreImg}^H_{\mathcal{A}}(\lambda)$
$k \leftarrow \mathsf{Kg}(\lambda)$
$x \leftarrow \{0,1\}^*$
$x' \leftarrow \mathcal{A}(k, H_k(x))$
Return 1 *iff* $H_k(x) = H_k(x')$.

Experiment $\mathsf{SecPre}^{\mathcal{H}}_{\mathcal{A}}(\lambda)$
$k \leftarrow \mathsf{Kg}(\lambda)$
$x \leftarrow \{0,1\}^*$
$x' \leftarrow \mathcal{A}(k, x)$
Return 1 *iff* $x \neq x'$
and $H_k(x) = H_k(x')$.

The probability is taken over all coin tosses of Kg *and* $\mathcal{A}$*.*

Note that the value k specifying the hash function H_k from the hash family is called key, but usually the key is not kept secret. There is an exception if we consider message authentication codes. In other words, a family of hash functions does not fulfill the security properties mentioned above only due to the secrecy of the key.
It is well known and straightforward to prove that any hash function which is collision resistant is also second pre-image resistant. That is, when designing a family of hash functions, one should aim for collision resistance and pre-image resistance.
Constructing hash functions is a hot topic. Specifically, it is desired to have hash functions whose design is based on a difficult mathematical problem and thus whose security follows rigorous mathematical proofs and formal reduction. Such functions are called provably secure hash functions and only a few examples of them were proposed in literature like SWIFFT [21], VSH [13] and ECHO [11].

Error-correcting codes. A linear code of length n and rank k is a linear subspace $\mathcal{C}$ with dimension k of the vector space $\mathbb{F}_q^n$ where $\mathbb{F}_q$ is the finite field with q elements. The elements of $\mathbb{F}_q^n$ (resp. $\mathcal{C}$) are called words (resp. codewords). The Hamming weight of a word x, denoted by $wt(x)$, is the number of its non-zero entries. A parity check matrix H of $\mathcal{C}$ is defined by $H \cdot x^T = 0$, $\forall x \in \mathcal{C}$. In this paper, we take $q = 2$.

Definition 2 (Regular word). *A regular word of length* n *and weight* w *is a codeword consisting of* w *blocks of length* n/w*, each has exactly one non-zero entry.*

Definition 3 (2-Regular word). *A 2-regular word is defined as a sum of two regular words. It is of length n and weight less that equal to $2w$.*

The security of code-based cryptosystems is based on the hardness of several coding theory problems. The most relevant in our context are stated in the following.

Definition 4 (Binary Syndrome Decoding (SD) problem). *Given a binary $s \times n$ matrix H, a binary vector $y \in \mathbb{F}_2^s$ and an integer $w > 0$, find a word $x \in \mathbb{F}_2^n$ of weight $wt(x) = w$, such that $H \cdot x^T = y$.*

This problem is proven NP-complete in [3]. In [1] two further problems related to SD have been shown to be NP-complete. They can be stated as follows.

Definition 5 (Regular Syndrome Decoding (RSD) problem). *Given a binary $s \times n$ matrix H, a binary vector $y \in \mathbb{F}_2^s$ and an integer $w > 0$, find a regular word $x \in \mathbb{F}_2^n$, $x \neq 0^n$, of weight $wt(x) = w$, such that $H \cdot x^T = y$.*

Definition 6 (2-Regular Null Syndrome Decoding (2-RNSD) problem). *Given a binary $s \times n$ matrix H and an integer $w > 0$, find a 2-regular word $x \in \mathbb{F}_2^n$ of weight $\leq 2w$, such that $H \cdot x^T = 0$.*

Throughout this paper, we will denote $\mathrm{SD}(n, s, w)$ and $\mathrm{RSD}(n, s, w)$ to indicate instances of the above problems with parameters (n, s, w).

3 Related Works

In this section, we briefly provide a description of the main ingredients that we need to design our family of hash functions: the FSB hash function [1] and the sponge construction [18].

FSB hash functions. The Fast Syndrome-Based Hash Functions (FSB) were first introduced in 2003 by Daniel Augot et al. [1] and improved in 2007 [16]. The FSB follows the iterative Merkle-Damgård design principle [22,15] based on the compression function $\mathcal{F}$ defined by

$$\begin{aligned} \mathcal{F} : \mathbb{F}_2^s &\to \mathbb{F}_2^r \\ x &\mapsto \mathcal{F}(x) = H \cdot \varphi_{n,w}(x)^T, \end{aligned}$$

where H is a random binary matrix of size $r \times n$ and the mapping $x \mapsto \varphi_{n,w}(x)$, called a regular encoder, is an encoding algorithm, which takes an s-bit and returns a regular word of length n and weight w. In each round, the input of the compression function $\mathcal{F}$ consists of s-bit string, which is a concatenation of r bits taken from the output of the previous round and $s - r$ bits taken from the message to be hashed. The use of this encoder speeds up the computing of the vector-matrix multiplication, since this process is equivalent to XORing w columns of length r from the matrix H. In order to obtain a smaller hash size the whirlpool-hash function [2] is applied on the pre-final hash value.

Remark 1. Recently and just after finishing this work, we have come to know that a new construction based on the FSB hash family due to Bernstein et al. [8] has been proposed. Their proposal seems to be more efficient than the FSB SHA-3 proposal.

Sponge construction. This construction was presented by Bertoni et al. [18] in 2007. It represents a new way to build hash functions from a random transformation/permutation, denoted by $\mathcal{S}$, operating on states of length $s = r + c$ bits. The parameter r is called the rate, c the capacity, and s the width. The initial state is initialized to zero. The first r (resp. the last c bits) of a state is called the outer part (resp. the inner part (or inner state)) of the construction. Let x denote the message to be hashed. After padding x, the hashing process consists of two phases: the absorbing phase followed by the squeezing phase. During the absorbing phase, only the outer part of the state is combined with a r-bit message block using the bitwise XOR-operation. The result is then fed through $\mathcal{S}$. After processing all blocks of x the squeezing phase follows. In this phase only the outer parts are returned as intermediate hash values, interleaved with applications of $\mathcal{S}$. The number of hash blocks is chosen at will by the user.

4 Our Construction: S-FSB

In this section, we present a variant of the FSB hash function, called *Sponge Fast Syndrome-Based* hash function (in short S-FSB). We will use the same notations as in the previous section and define five positive integers n, w ,s, r and c such that the ratio n/w is a power of 2, and $s = r + c = w \log_2(n/w)$.

4.1 Description of S-FSB

The main idea behind our proposal is to use the sponge construction [18] of mode of operation rather than the Merkle-Damgard mode [22,15] of operation used in FSB. The S-FSB is based on the FSB transformation introduced in [19] to design the SYND stream cipher. This transformation, denoted here by $\mathcal{T}$, is defined by:

$$\begin{aligned} \mathcal{T} : \mathbb{F}_2^s &\to \mathbb{F}_2^s \\ x &\mapsto \mathcal{T}(x) = H \cdot \varphi_{n,w}(x)^T, \end{aligned}$$

Where H is a random binary matrix of size $s \times n$ and the mapping $x \mapsto \varphi_{n,w}(x)$ is a regular encoding algorithm as in FSB. For plugging this transformation into the sponge construction, we take s width, r the rate, and c the capacity (see section 3) such that $s = r + c$ (see section 3).

As illustrated in Figure 1, absorbing each r-bit message block m_i is performed as follows. The block m_i is first combined with the outer part of the current state (the previous output of $\mathcal{T}$) using the bitwise XOR operation. The result is then encoded into a regular word y of length n and weight w by applying the regular encoder $\varphi_{n,w}$. Finally, the multiplication of y by the transpose of H (denoted H^T) is performed to get the next state. When all message blocks are processed, the construction switches to the squeezing step as in the sponge construction to output the first r bits of the state as hash blocks. As in the sponge construction, those blocks form the pre-final hash value, which are extracted to get the final hash value of length l.

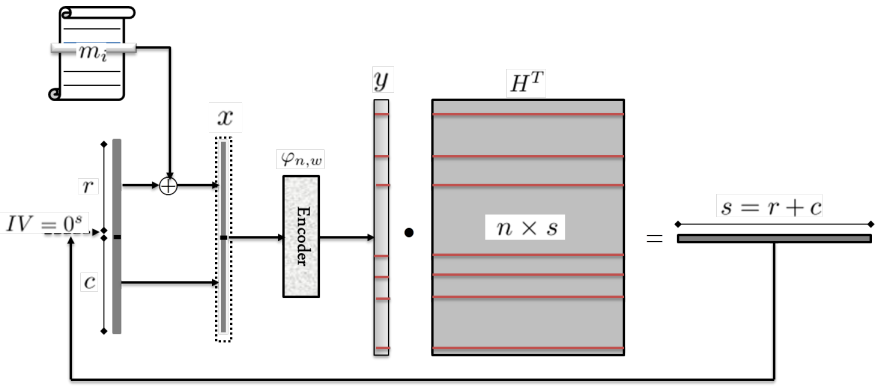

Fig. 1: Absorbing step of S-FSB hash function

The performance of our proposal depends directly on the number of bitwise XOR operations computed at each round to treat the r bits of one message block. That is, one needs first r XORs for the bitwise addition and then s XORs of w columns of the matrix H. This result to $r + sw$ binary XOR-operations. Since the number of bits of each message block is r, the number of expected binary XORs (denoted by $\mathcal{N}_{xor}$) in average for each message input bit is:

$$\mathcal{N}_{xor}(n, w, r, c) = \frac{r + (r + c)w}{r}.$$

where $r + c = s = w \log_2(n/w)$. This results to

$$\mathcal{N}_{xor}(n, w, r, c) = 1 + \frac{w^2 \log_2(n/w)}{r} \quad (1)$$

This quantity is the main measure to estimate the theoretical performance of our proposal.

5 Security Analysis

In this section, we consider the security of S-FSB. We will show how the security of S-FSB is reduced to the security of the syndrome decoding problem. More precisely, finding of pre-images (resp. collisions) is reduced to regular syndrome decoding problem (resp. 2-regular null syndrome decoding problem). Furthermore, we analyze the security in terms of current best known algorithms for solving these two problems.

5.1 Theoretical Security

In a cryptography environment, we require more properties of a hash function than just compressing the input value into a short bit string. In particular, there are three basic security requirements that a cryptographic hash function at least should fulfill. Namely, this is collision and (second) pre-image resistance. These requirements are defined in Definition 1.
In order to analyze the security of sponge-based hash functions, we need to introduce the following definitions to understand how generic attacks work against our proposal.

Definition 7 (Absorbing function). *Let $\mathcal{S}$ be a sponge function. The absorbing function $\mathtt{abs}[\cdot]$, takes as input a padded message x of length multiple of r and returns the value of the state e obtained after absorbing x, i.e. $\mathtt{abs}[x] = e$.*

Definition 8 (Path). *Let $\mathcal{S}$ be a sponge function. An input x is called a path to the state e if $\mathtt{abs}[x] = e$.*

Definition 9 (Squeezing function). *Let $\mathcal{S}$ be a sponge function. The squeezing function, denoted by $\mathtt{sqz}[\cdot]$, takes as input a state e given at the beginning of the squeezing step and returns an l-bit string Z_l the output truncated to l bits of $\mathcal{S}$.*

We will denote by e_c the inner state of a state e. i.e. the last c bits of state e.

Definition 10 (Output binding). *Given an arbitrary string Z. Output binding is to find a state e such that $\mathtt{sqz}[e] = Z$.*

Definition 11 (State Collision). *Let $\mathcal{S}$ be a sponge function. A state collision is a pair of two different paths $x, x' \in \mathbb{F}_2^*$ such that $\mathtt{abs}[x] = \mathtt{abs}[x']$.*

Definition 12 (Inner Collision). *Let $\mathcal{S}$ be a sponge function. An inner collision is a pair of two distinct paths $x, x' \in \mathbb{F}_2^*$ resulting in the same inner part, i.e. $\mathtt{abs}[x]_c = \mathtt{abs}[x']_c$.*

It is easy to check that a state collision is an inner collision, since any two distinct paths $x, x' \in \mathbb{F}_2^*$ such that $\mathtt{abs}[x] = \mathtt{abs}[x']$ leads to $\mathtt{abs}[x]_c = \mathtt{abs}[x']_c$. However, the converse does not hold.

Notions for Security. In order to prove collision resistance of our proposed hash function family S-FSB, we need to show that experiment $\mathsf{Col}_{\mathcal{A}}^{H}$ from Definition 1 evaluates to 1 only with negligible probability where $H : \{0,1\}^* \rightarrow \{0,1\}^l$ is the S-FSB hash family. Let $\mathsf{Adv}_{H,\lambda}^{Col}$ denote the maximum probability over any PPT algorithm $\mathcal{A}$ to compute a collision as defined in experiment $\mathsf{Col}_{\mathcal{A}}^{H}(\lambda)$, i.e. $\mathsf{Adv}_{H,\lambda}^{Col} = \max_{PPT\mathcal{A}}\{\mathrm{Prob}\left[\mathsf{Col}_{\mathcal{A}}^{H}(\lambda) = 1\right]\}$.
(Second) Pre-image resistance is defined analogously denoted by $\mathsf{Adv}_{H,\lambda}^{PreImg}$ (resp. $\mathsf{Adv}_{H,\lambda}^{SecPre}$). Furthermore, we reduce to security of our scheme to the syndrome problem (see Definition 4) which requires to specify its hardness. We denote $\mathsf{Adv}^{SD}(n, s, w)$ the maximum probability over any PPT algorithm $\mathcal{A}$ to solve the syndrome decoding problem with parameters (n, s, w).

5.1.1 Collision Resistance. Security properties of hash functions build upon the sponge methodology are scrutinized against generic attacks assuming random transformation $\mathcal{S}$. In terms of collision resistance the following statements are elaborated. An inner collision can easily (i.e. in polynomial time) transformed into a state collision. Assume x, x' are given inputs to a state collision, i.e. $\mathtt{abs}[x] = \mathtt{abs}[x']$. Then choose $p \in \{0,1\}^*$ randomly and set $m := x||p$ and $m' := x'||p$. Obviously, m, m' lead to a collision in the output of the hash function since $\mathtt{sqz}[\mathtt{abs}[x||p]] = \mathtt{sqz}[\mathtt{abs}[x'||p]]$. Therefore, shown in [18], assuming a random sponge the workload of generating a collision in the output of the hash function is of the order $\min\{2^{(c+3)/2}, 2^{(l+3)/2}\}$. Due to space limitations, we give here a short intuition how the security proof for collision resistance works and refer interested reader to the full version of this paper.
In order to produce collision for the S-FSB hash family where the sponge transformation is defined as $\mathcal{T} : \mathbb{F}_2^s \to \mathbb{F}_2^s;\ x \mapsto \mathcal{T}(x) = H \cdot \varphi_{n,w}(x)^T$ where H is a random binary matrix of size $s \times n$, we require to analyze the complexity of generating an inner collision. Since the capacity of a state is fixed (e.g. by $IV = 0$) in the beginning of the absorption phase, it suffices for an adversary to find two r bit values m, m' such that $\mathtt{abs}[m]_c = \mathtt{abs}[m']_c$. The complexity of generating collisions for transformation $\mathcal{T}$ is equivalent to $SD(n, s, 2w)$, shown in the security proof of FSB [16]. At the same time, for an inner collision, one needs to solve $SD(n_r, c, 2w_r)$ where $n = n_r + n_c$ and $w = w_r + w_c$ are the corresponding columns of H and weights of the input regular word belonging to the first r bits and the last c bits, respectively.
This leads to the following proposition.

Proposition 1 (Collision Resistance). *Let h be an S-FSB$_{(n,w,r,c)}$ hash function scheme instantiated with parameters (n, w, r, c) where $n = n_r + n_c$ and $w = w_r + w_c$. Then, we have*

$$\mathsf{Adv}^{Col}_{h,(n,w,r,c)} \leq \mathsf{Adv}^{SD}(n_r, c, 2w_r) + \mathsf{Adv}^{SD}(n, s, 2w) + 2^{-(c+3)/2} + 2^{-(l+3)/2}.$$

Proof. Proof is given in the full version.

5.1.2 (Second) Pre-image Resistance Unfortunately, due to space limitations, we only provide our results of (second) pre-image resistance and refer the reader to the full version of the paper for the corresponding proof.

Proposition 2 ((Second) Pre-Image Resistance). *Let h be an S-FSB$_{(n,w,r,c)}$ hash function scheme instantiated with parameters (n, w, r, c) where $n = n_r + n_c$ and $w = w_r + w_c$. Then, we have*

$$\mathsf{Adv}^{PreImg}_{h,(n,w,r,c)} \leq \mathsf{Adv}^{SD}(n_r, c, w_r) + \mathsf{Adv}^{SD}(n, l, w) + 2^{-l} + 2^{-(c-1)}$$

,and

$$\mathsf{Adv}^{SecPre}_{h,(n,w,r,c)} \leq m \cdot (\mathsf{Adv}^{SD}(n_r, c, w_r) + 2^{-c/2})$$

where m is the path length of a given pre-image.

Proof. Proof is given in the full version.

5.2 Practical Security

In practice, to assess the security of our scheme regarding the collision and (second) preimage resistance, we have to identify all known applicable attacks and to estimate the minimal complexities required to execute these attacks. As far as we know, there exist three kind of attacks: Information Set Decoding (ISD), Generalized Birthday Attack (GBA).

Information Set Decoding (ISD). ISD attacks are probabilistic algorithms for solving the SD problem. The main idea behind an ISD attack to find a valid set (information set) of $k = n - s$ (k is the dimension and n the length of the code) positions among the n positions. This set is valid if the SD problem has a solution whose support does not meet the chosen k positions. To check the validity of this set, one has to perform the Gaussian elimination of an $s \times s$ submatrix of the parity check matrix H of size $s \times n$, and thereby the whole complexity of this algorithm is expressed as $G(s)/P(n, s, w)$, where $G(s)$ is the cost of the Gaussian elimination and $P(n, s, w)$ the probability to find a valid information set.

Let $P_r(n,s,w)$ be the probability that a given information set is valid for one given solution of RSD. Let denote by $N_r(n,s,w)$ the expected number of solution of RSD. As stated in [1], the probability $P(n,s,w)$ can approximated by $P(n,s,w) = P_r(n,s,w) \times N_r(n,s,w)$. Since there exist $\binom{n}{w}^w$ regular words, then the average number of solutions of RSD is

$$N_r(n,s,w) = \frac{\binom{n}{w}^w}{2^s}.$$

In our setting, we have $s = w\log 2(n/w)$. This results in $N_r(n,s,w) = \frac{\binom{n}{w}^w}{\binom{n}{w}^w} = 1$. That means, we have only one solution to RSD, on average. Furthermore, as shown in [1], the $P_r(n,s,w)$ is given by

$$P_r(n,s,w) = \binom{s}{n}^w = \left(\frac{\log_2(n/w)}{n/w}\right)^w$$

If we set $\log_2(n/w) = \beta$, for some integers β, then the final probability of selecting a valid set to invert RSD equals to:

$$P(n,s,w) = P_r(n,s,w) \times N_r(n,s,w) = \left(\frac{\beta}{2^\beta}\right)^w \text{ with } \beta = \log_2(n/w). \tag{2}$$

To estimate the cost of finding collisions, we have to evaluate the complexity of solving the 2-RNSD problem stated above. This can be done in the same way as in [1]. We compute the number of two-regulars words, then we multiply it by the probability of the validity, to get the total probability of choosing a valid set. This probability, denoted by P_I, is given by:

$$P_I(n,s,w) = \binom{w}{n}^w \left[\binom{\log_2(n/w)}{2} + 1\right]^w$$

For simplicity, we can assume that $\beta \geq 2$. So, we get an upper bound for this probability, denoted by P_C, which is equal to:

$$P_C(n,s,w) = \left(\frac{\beta^2}{2^{\beta+1}}\right)^w \text{ with } \beta = \log_2(n/w). \tag{3}$$

From the equation (3), we conclude that the probability for a random information set to be valid in case of collisions search is larger by a factor $(\frac{\beta}{2})^w$ compared to the probability for a random information set to be valid in case of finding preimages, where $\beta = \log_2(n/w)$.

In practice, there exists a lower bound for information set decoding attacks, presented in [6]. We will use these bounds to estimate the security of our scheme.

Generalized Birthday Attack (GBA). The GBA algorithm is due to Wagner [25] and was introduced to cryptanalyze the FSB hash function by Coron et al. [14].
We now describe the attack from Matthieu and Sendrier [17], which relies on the Generalized Birthday Problem introduced by Wagner [25]. The basic idea behind this algorithm is, for a given integer α, to find a set of indexes $\mathcal{I} = \{1, 2, \cdots, 2^\alpha\}$ verifying

$$\bigoplus_{i \in \mathcal{I}} H_i = 0.$$

To find this set $\mathcal{I}$, one has to compile 2^α lists of $2^{\frac{s}{\alpha+1}}$ elements containing distinct columns of the matrix H of size $s \times n$. These lists are then pairwise combined to get $2^{\alpha-1}$ lists of XORs of 2 columns of H. In the resulting lists, only 2 columns starting with $\frac{s}{\alpha+1}$ zeros are kept, instead of all the possible columns. Then, the new lists are pairwise merged to obtain $2^{\alpha-2}$ lists of XORs of 4 columns of H. Only 4 columns of H starting with $2\frac{s}{\alpha+1}$ zeros, are kept. This process will be continued, until only two lists are left. These two lists will contain $2^{\frac{s}{\alpha+1}}$ XORs of $2^{\alpha-1}$ columns of H having $(\alpha-1)\frac{s}{\alpha+1}$ zeros at the beginning. After that, the standard birthday algorithm can be applied to get one solution. Since all lists treated above, have the same size $\frac{s}{\alpha+1}$, the complexity of GBA is at least in $O\left(\frac{s}{\alpha+1}2^{\frac{s}{\alpha+1}})\right)$.

As we can see in this algorithm, the number of XORed columns was a power of 2. However, this does not hold in general because the weight w can be any number. So if w is not a power of 2, one can modify the above algorithm such that one can back in the general case of GBA by imposing the following condition on α: $\frac{1}{2^\alpha}\binom{n}{\frac{2w}{2^\alpha}} \geq 2^{\frac{s-\alpha}{\alpha}}$ (see [17] for more details). This condition can be rewritten as:

$$\binom{2^\beta w}{2^{(1-\alpha)}w} \geq 2^{\beta w+\alpha(\alpha-1)} \tag{4}$$

where $\log_2(n/w) = \beta$. In this case, one gets a lower bound of the cost of solving an instance SD problem with parameters (n, s, w) as follows:

$$\left(\frac{w\beta}{\alpha} - 1\right) 2^{\frac{w\beta}{\alpha}-1}. \tag{5}$$

As we can see, for fixed weight w, this complexity is an increasing function in n. So, to avoid the GBA attack, we have to choose large n.

Remarks:

- In [5] an implementation of GBA is presented against the compression function of FSB. This implementation includes two techniques introduced in [4] in order to mount GBA on computers, which do not have enough storage capacity to hold all list entries. However, the complexity of this attack is still exponential. Since our scheme is based on the FSB compression function, we claim that our proposal is secure against this implementation.
- It was shown in [20] that the sponge-based hash functions can be attacked by slide attacks. This kind of attacks was introduced in [10] by Biryukov et.al for cryptanalyzing iterative block ciphers. For attacking a sponge-like construction, the self-similarity issue can be exploited, meaning that all the blank rounds behave identically. As noted in [20], a simple defense against slide attacks consists in adding nonzero constant just before running the blank rounds. This can be achieved by a convenient padding such that the last block of the message is different from null vector. That is exactly, what we are used in our construction. Therefore, our proposal is secure against slide attacks.
- In [24], the so-called linearization attack (LA) was proposed against FSB to find collisions. The key idea is to reduce the problem of finding collisions to a linear algebra problem that can be solved in polynomial time, when the ratio s/w is up to 2. Furthermore, as shown in [24], this attack can still be applied if $s > w$. It can be extended even to $s > 2w$ with complexity $O(s^3 \left(\frac{3}{4}\right)^{s-2w})$. So, to avoid the LA attack, we have to choose $s > 2w$.
- Most recently a new variant of ISD algorithm [7] was presented for estimating the hardness of the 2-Regular Null Syndrome Decoding problem (2-RNSD). This algorithm runs faster than the lower bounds given in [17]. The parameters we propose in the next section are chosen to resist this attack as well.

6 Proposed Parameters

When selecting parameters for S-FSB, we have to look for parameters providing the desired security with least processing cost required to hash one bit of the message. As mentioned in Section 4, this cost can be measured using formula (1), which is expressed as a function depending on the parameters set (n, w, r, c). This function is defined by

$$\mathcal{N}_{xor}(n, w, r, c) = 1 + \frac{w^2 \log_2(n/w)}{r} = 1 + \frac{ws}{r} = 1 + \frac{w(r+c)}{r} = 1 + w\left(1 + \frac{c}{r}\right) \tag{6}$$

We observe that for increasing values of c, this function is an implicitly increasing quantity in w and n. So, if we want to have a good performance, then we have to choose small values of c (as small as possible) and select w and n such that the value of r are large. But from security point of view, we should choose s greater than $2w+1$ to withstand the linearization attack mentioned earlier. Furthermore, to avoid inner and outer collisions, the running time of solving instances of RSD and 2-RNSD with parameters (n, s, w) and (n_r, c, w_r) according the best known collision attack, must be larger than the desired security.

Starting from those conditions, we propose three parameter sets (n, s, w, c) that provide different security levels. Those sets of parameters are presented in Table 1 together with the corresponding numbers of XORs and the complexities of the ISD and GBA attack.

						Preimage		Collision	
Hash size	n	s	w	c	$\mathcal{N}_{xor}$	GBA	ISD	GBA	ISD
160	$3 \cdot 2^{19}$	384	24	240	64.0	2^{130}	2^{99}	2^{86}	2^{91}
224	$17 \cdot 2^{17}$	544	34	336	88.9	2^{150}	2^{144}	2^{114}	2^{122}
256	$39 \cdot 2^{17}$	624	39	296	90.5	2^{246}	2^{172}	2^{129}	2^{148}

Table 1: Proposed paramaters for S-FSB

7 Performance Evaluation

S-FSB has been implemented on a 2.53 GHz Pentium Core2 Duo, running Linux (Ubuntu 10.04) 32 Bit with 6MB of cache and 4GB of RAM. The C compiler is GCC, version 4.4.3 with -O3 optimization. In our implementation, we used truncated quasi-cyclic codes as in FSB. We propose three versions of S-FSB of hash size 160, 224, and 256 bits. The performance of these versions is reported in Table 2. This performance was measured on a message of size 1000 MB. The file hash time in the third row was measured by repeated calls to the `clock()` function to get the current millisecond clock value and subtracted the stop time from the start time. The number of samples we performed is about one million. To get the speed expressed in cycles per bytes, we multiplied the measured hash time by the CPU frequency and divided the result by the file size in bytes.

Hash size (bits)	File size (MB)	File hash time (s)	Speed (cpb)
160	1000	66.90	≈ 160
224	1000	84.48	≈ 201
256	1000	75.63	≈ 183

Table 2: Performance of S-FSB on a 2.53 GHz Core2 Duo processor.

In order to compare our results with those of FSB SHA-3 proposal[1], we ran the C-code of FSB on the same desktop and we obtained the results presented in Table 3. As we can see, the S-FSB is more efficient than FSB by a factor of 1.44 (30%). Despite this improvement, the S-FSB hash function remains slower than the existing hash functions like the SHA-2 family.

References

1. D. Augot, M. Finiasz, and N. Sendrier. A Family of Fast Syndrome Based Cryptographic Hash Functions. In E. Dawson and S. Vaudenay, editors, *Mycrypt 2005*, volume 3715, pages 64–83. Springer, 2005.
2. P. S. L. M. Barreto and V. Rijmen. Whirlpool. Seventh hash-function of ISO/IEC 10118-3:2004, 2004.

[1] http://www-rocq.inria.fr/secret/CBCrypto/index.php?pg=fsb.

Hash size (bits)	File size (MB)	File hash time (s)	Speed (cpb)
160	1000	87.76	≈ 212
224	1000	102.99	≈ 248
256	1000	109.38	≈ 264

Table 3: Performance of FSB SHA-3 proposal on a 2.53 GHz Core2 Duo processor.

3. E. Berlekamp, R. McEliece, and H. van Tilborg. On the inherent intractability of certain coding problems. *IEEE Transactions on Information Theory*, 24(2):384–386, May 1978.
4. D. J. Bernstein. Better price-performance ratios for generalized birthday attacks, 2007.
5. D. J. Bernstein, T. Lange, R. Niederhagen, C. Peters, and P. Schwabe. FSBDay: Implementing wagner's generalized birthday attack against the SHA-3 candidate FSB, 2009.
6. D. J. Bernstein, T. Lange, and C. Peters. Ball-Collision Decoding. Cryptology ePrint Archive, Report 2010/585, 2010. `http://eprint.iacr.org/`.
7. D. J. Bernstein, T. Lange, C. Peters, and P. Schwabe. Faster 2-regular information-set decoding, 2011.
8. D.J. Bernstein, T.Lange, C. Peters, and P. Schwabe. Really fast syndrome-based hashing. Cryptology ePrint Archive, Report 2011/074, 2011. `http://eprint.iacr.org/`.
9. G. Bertoni, J. Daemen, M. Peeters, and G. Van Assche. Keccak specifications. Submission to NIST (Round 2), 2009.
10. A. Biryukov and D. Wagner. Slide attacks. In *FSE*, volume 1636 of *LNCS*, pages 245–259. Springer, 1999.
11. D. R. L. Brown, A. Antipa, M. Campagna, and R. Struik. Ecoh: the elliptic curve only hash. Submission to NIST, 2008.
12. Ch. De Cannière and C. Rechberger. Finding sha-1 characteristics: General results and applications. In Xuejia Lai and Kefei Chen, editors, *ASIACRYPT*, volume 4284 of *LNCS*, pages 1–20. Springer, 2006.
13. S. Contini, A. K. Lenstra, and R. Steinfeld. Vsh, an efficient and provable collision-resistant hash function. In *LNCS*, pages 165–182. Springer, 2006.
14. J.-S. Coron and A. Joux. Cryptanalysis of a provably secure cryptographic hash function. Cryptology ePrint Archive, Report 2004/013, 2004. `http://eprint.iacr.org/`.
15. I. Damgård. A Design Principle for Hash Functions. In Gilles Brassard, editor, *Advances in Cryptology – CRYPTO '89, Proc.*, volume 435 of *LNCS*, pages 416–427. Springer, 1990.
16. M. Finiasz, P. Gaborit, and N. Sendrier. Improved fast syndrome based cryptographic hash functions. In V. Rijmen, editor, *ECRYPT Hash Workshop 2007*, 2007.
17. M. Finiasz and N. Sendrier. Security Bounds for the Design of Code-based Cryptosystems. In Mitsuru Matsui, editor, *Advances in Cryptology - ASIACRYPT 2009*, number 5912 in LNCS, pages 88–105. Springer, 2009.
18. M. Peeters G. Bertoni, J. Daemen and G. Van Assche. Sponge Functions. In *ECRYPT Hash Workshop 2007*, 2007.
19. Ph. Gaborit, C. Laudaroux, and N. Sendrier. Synd: a very fast code-based cipher stream with a security reduction. In *IEEE Conference, ISIT'07*, pages 186–190, Nice, France, July 2007.
20. M. Gorski, S. Lucks, and T. Peyrin. Slide attacks on a class of hash functions. In *ASIACRYPT '08: Proc. of the 14th International Conference on the Theory and Application of Cryptology and Information Security*, pages 143–160. Springer, 2008.
21. V. Lyubashevsky, D. Micciancio, C. Peikert, and A. Rosen. Swifft: A modest proposal for fft hashing. pages 54–72, 2008.
22. R. C. Merkle. One Way Hash Functions and DES. In Gilles Brassard, editor, *Advances in Cryptology – CRYPTO '89, Proc.*, volume 435 of *LNCS*, pages 428–446. Springer, 1990.
23. National Institute of Standards and Technology (NIST). *Secure Hash Standard*, October 2008.
24. M.-J. O. Saarinen. Linearization attacks against syndrome based hashes. In K. Srinathan, C. Pandu Rangan, and Moti Yung, editors, *INDOCRYPT*, volume 4859 of *LNCS*, pages 1–9. Springer, 2007.
25. D. Wagner. A generalized birthday problem. In Moti Yung, editor, *CRYPTO*, volume 2442 of *LNCS*. Springer, 2002.
26. X. Wang, Y. L. Yin, and H. Yu. Finding collisions in the full sha-1. In *In Proceedings of Crypto*, pages 17–36. Springer, 2005.