Interpreting Hash Function Security Proofs

Juraj Šarinay*

EPFL IC LACAL, Station 14, CH-1015 Lausanne, Switzerland
juraj.sarinay@a3.epfl.ch

Abstract. We provide a concrete security treatment of several "provably secure" hash functions. Interpreting arguments behind MQ-HASH, FSB, SWIFFTX and VSH we identify similar lines of reasoning. We aim to formulate the main security claims in a language closer to that of attacks. We evaluate designers' claims of provable security and quantify them more precisely, deriving "second order" bounds on bounds. While the authors of FSB, MQ-HASH and SWIFFT(X) prove existence of non-trivial lower bounds on security, we show that the quantification of the bounds limits the practical significance of the proofs.

Keywords: hash functions, security bounds, provable reducibility

1 Introduction

A hash function is a mapping that on input a string of arbitrary length outputs a digest of fixed size. Such functions are among the most basic building blocks used in cryptographic schemes. Several traditional hash functions are now considered broken [44, 43, 39] and for a few years there has been an urgent need for new ideas. The NIST search for a new standard attracted over sixty submissions [29]. As the competition entered Round 2 in July 2009, the number of participants was reduced to fifteen.

Confidence in the security of hash functions relies traditionally on cryptanalysis. As an alternative, one may try to prove security properties. With a valid security proof at our disposal, we would not need to consider attacks any more.

Several "provably secure" hash functions appeared recently [8, 12, 31, 11, 26, 1, 16, 15]. The language of the proofs is often incompatible with the view of a practitioner. Cryptanalysts normally speak of rather precise time estimates for programs running on real-world hardware. Security proofs should use similar language to provide lower bounds on the effort needed to break a function. We choose to speak of proofs in "real life" and require security to be quantified. We do not view security as a property, but rather as a measure.

A security proof is a *conditional* statement relying on a hardness assumption. Confidence is "transferred" from the (hopefully more basic) hard problem to the hash function. There needs to be some level of confidence to begin with, such assumptions need to be selected very carefully. Examples of "provably secure" functions based on false assumptions include the Zémor-Tillich, LPS and

* Supported by a grant of the Swiss National Science Foundation, 200021-116712.

Morgenstern hash functions proposed in [40, 11, 31, 33] and broken in [41, 32, 20]. This paper only deals with functions believed not to be completely broken. In addition we treat assumptions *literally* as assumptions, i.e. we look at what is implied by them without considering their validity in detail. We concentrate on the structure of proofs and their use in security assessment.

We interpret "provable security" arguments behind MQ-HASH [8], FSB [15], SWIFFTX [1] and VSH [12]. Where possible, we take a concrete viewpoint and carefully follow any comparisons between provable bounds and cost of attacks done by the designers. It turns out that the security is sometimes bounded in a rather complex way, failing to provide a proof.

In case of MQ-HASH and FSB we point out gaps in reasoning. No inconsistencies are discovered in the arguments supporting SWIFFT. We observe that although the improved design of SWIFFTX prevents known attacks, the predecessor is as good in terms of proofs.[1] Very Smooth Hash does provide a concrete lower bound on collision resistance. This is only true of some variants.

Related Work The general limitations of security proofs in cryptology were discussed by Koblitz and Menezes [22, 21]. A concrete analysis of "provable" claims about a particular stream cipher appeared in [45]. This paper also questions security proofs, looking at hash functions that have not been considered from this perspective.

Block cipher based hash functions enjoy well established provable security properties [34, 9, 38]. Such constructions assume access to an ideal cryptographic primitive, e.g. an ideal cipher. The hardness assumptions and security claims are thus of a different nature compared to the cases considered in this paper. In addition, the framework normally considers adversaries with oracle access to the idealized building block, hence a straightforward comparison to practical attacks is not possible.

All functions considered in this paper were previously cryptanalyzed. This includes work on FSB [13, 7], MQ-HASH [4], SWIFFT [10] and VSH [36]. Interestingly, the papers do not relate attacks to the original security proofs.

2 Preliminaries

The functions considered in this paper follow the Merkle-Damgård construction, or a variant thereof [14]. We limit ourselves to *pre-images* and *collisions*. In our analysis we can therefore stick to fixed-length compression functions. Let $\{H_k\}_{k\in K}$ be a finite family of compression functions $H_k : \{0,1\}^m \to \{0,1\}^n$. Most basic security properties are tied to the hardness of the following two tasks:

Find a pre-image Given a random $k \in K$ and an element y in the range of H_k compute x such that $H_k(x) = y$.

[1] This observation was implicit in [1].

Find a collision Given a random $k \in K$ compute $x \neq x'$ such that $H_k(x) = H_k(x')$.

Instead of a single fixed function we work with a family of functions parametrized by k, following the designers of all four functions analyzed in Section 3. This is a standard way towards arguments on collision resistance.[2] Prove the security for *random* members of the function family, then pick a single member at random. Such proofs are believed to provide some confidence in security, although efficient algorithms breaking the fixed function *do* exist.

"Classical" definitions prescribe ideal measures of hardness for the two problems. Due to a generic attack, a pre-image can be found after approximately 2^n evaluations of H_k. By the birthday paradox, collisions can be found after about $2^{n/2}$ evaluations of H_k. Hence the usual requirement is "n bit" pre-image resistance and "$n/2$ bit" collision resistance. Rather than following this ambitious goal, let us look for *any* measure of hardness.

Cost of Attacks Fix a hash function family $\{H_k\}_{k \in K}$. Let elements of a set S represent computational cost. The precise nature of S will vary between families of functions. There are several ways to measure cost of attacks on a cryptographic primitive. Practical examples include hardware cost, processor cycle count, memory. On the more theoretical side, one can measure advantage as a function of runtime, expected runtime, circuit size, etc. For the functions we examine, choices of S are implicit in the statements of theorems on security. In order to allow different approaches to cost, we only require S to be a partially ordered set, i.e. equipped with a *reflexive, antisymmetric* and *transitive* relation '$\leq$'. This captures the minimal assumption that complexities can be compared and allows a lot of freedom in formalization of cost. While it might appear natural to require the ordering on S to be *linear* as well, it is not necessary for our purposes.[3] Let S contain a least element and a greatest element. The former corresponds to zero cost, the latter to cost that is considered too high to be relevant. In most cases, the elements of S will turn out to be numbers counting some basic operation, such as a bit operation or evaluation of H_k.

Let $\mathcal{A}_{\mathrm{Pr}}$ be the set of all probabilistic algorithms that take as input a random $k \in K$, random n-bit y and with non-zero probability output an m-bit string x, such that $H_k(x) = y$.

Similarly, let $\mathcal{A}_{\mathrm{Col}}$ be the set of all probabilistic algorithms that take as input a random $k \in K$ and with non-zero probability output two different m-bit strings x, x', such that $H_k(x) = H_k(x')$. For $A \in \mathcal{A}_{\mathrm{Pr}} \cup \mathcal{A}_{\mathrm{Col}}$ define $c(A) \in S$ to be the expected cost of A solving the respective challenge.

The sets $\mathcal{A}_{\mathrm{Pr}}$ and $\mathcal{A}_{\mathrm{Col}}$ contain all the possible attacks on the security of $\{H_k\}_{k \in K}$, in particular the respective generic attacks.

---

[2] For comments and a different approach to the "foundations-of-hashing" dilemma see [35].

[3] This extra freedom can even be desirable.

Bounding Security Let $\mathcal{A}$ represent either of $\mathcal{A}_{\mathrm{Pr}}$ or $\mathcal{A}_{\mathrm{Col}}$. Define the following two main *types* of bounds on security:

- $p \in S$ is a bound of type **L** if for all $A \in \mathcal{A}$ the cost $c(A)$ is at least p.
- $q \in S$ is a bound of type **U** if for every $p \in S$ of type **L** it holds that $p \leq q$.

This captures the fact that every attack leads to an upper bound on security. If p is of type **L** and q of type **U**, then $p \leq q$. Note that we chose to define type **U** bounds with respect to **L** bounds, rather than relating them to attacks directly. It is immediate that $c(A)$ is of type **U** for any $A \in \mathcal{A}$. Yet our definitions allow $q \in S$ to be a **U** bound even if there is no $A \in \mathcal{A}$ such that $q = c(A)$. Similarly, it is possible that $p \in S$ is of type **L** and there is no $A \in \mathcal{A}$ such that $p = c(A)$.

We say $\{H_k\}_{k \in K}$ is *provably secure* if we possess a proof that $p \in S$ is of type **L**. The p is a *lower bound* on security against $\mathcal{A}$. A **U** type bound q is an *upper bound* on provable security.

While type **U** bounds are quite common, as will be shown in Section 3, designers do not often establish bounds of type **L**. To interpret certain results, we shall need the two following auxiliary bound types:

- $r \in S$ is a bound of type **lU**, if $r \leq t$ for t of type **U**. This means r is a lower bound on *some* upper bound on security.
- $s \in S$ is a bound of type **uL**, if $s \geq t$ for t of type **L**, i. e. s is an upper bound on *some* lower bound on security.

Every **L** bound is an **lU** bound and every **U** bound is an **uL** bound. Such implications are trivial, as *any* element of S is of type **lU** and **uL** simultaneously. The "weak" types **uL** and **lU** hence provide no direct useful information on provable security. Both are related to a bound t of one of the "useful" types **U** and **L**. Still the types **uL** and **lU** can carry partial information about the bound t if it is not quantified otherwise. For example, an **uL** bound tied to a particular reduction limits how good the reduction is and an **lU** bound tied to a particular attack algorithm limits how much damage the attack causes.

3 Results on Some Hash Functions

In this section we examine security proofs for hash functions and classify the bounds derived by their designers. To illustrate the proofs we follow a single concrete parameter choice from the original proposals. We follow original cost estimates and comparisons, i.e. adopt the implicit S and $c(\cdot)$.

The hardness assumption necessarily has the form of a lower bound. Ideally, security proofs would transform *any* attack on the function to an attack on the underlying hard problem, thereby establishing an **L** bound on security.

Criteria on Proofs We evaluate security proofs using two simple conditions:

- Is an **L** bound on security established and quantified?
- Does the security level claimed by designers match the proved **L** bound?

3.1 MQ-HASH

The MQ-HASH is a hash function built on the hardness of solving systems of multivariate quadratic equations over $\mathbb{Z}/2\mathbb{Z}$. It was introduced in [8].

Definition The compression function maps $m + n$ bit input to n bits. Let $r \geq m+n$, and f be an r-tuple of quadratic polynomials in $m+n$ variables over $\mathbb{Z}/2\mathbb{Z}$. Let g be an n-tuple of quadratic polynomials in r variables over the same field. The precise values proposed are $m = 32$, $n = 160$ and $r = 464$. This means f maps 192 bits to 464 bits and g maps 464 bits to 160 bits. Both f and g are to be selected uniformly at random. The compression function of MQ-HASH is the composition of f and g mapping 192 bits to 160 bits.

Hardness Assumption The designers assume that inverting random systems (of the type of) f and g is hard. For f, the hardness is quantified at $l_f = 2^{103.88}$ binary operations. By assumption, l_f is of type **L**. Because it is a complexity of an actual attack, it has type **U** as well. The implicitly present type **L** bound tied to g is not quantified. Denote it by l_g.

Security Claims The designers claim the function pre-image resistant proving the following:

Theorem 1. *Let T_f and T_g denote the time required to evaluate f, resp. g. Let A be an algorithm inverting (a random) $g \circ f$ in time T with probability ε. Then A can be either converted to an algorithm inverting g in time $T + T_f + T_g$ with probability ε or to an algorithm that can invert randomly chosen tuples of* 464 *quadratic polynomials in* 192 *variables that runs in time*

$$T' = \frac{128 \times 192^2}{\varepsilon^2}\left(T + T_f + 3T_g + \log\left(\frac{128 \times 192}{\varepsilon^2}\right) + 464 \times 192 + 2\right)$$

and succeeds with probability $\varepsilon/2$.

The theorem coupled with the hardness assumption(s) does indeed establish a lower bound on cost of any algorithm inverting $g \circ f$. If A inverts g, then $T + T_f + T_g \geq l_g$. This implies a lower bound $l_1 = l_g - T_f - T_g$ on T. If A leads to an algorithm inverting f, then $T' \geq l_f$ and $T \geq l_2$ for some l_2. This leads to a provable **L** bound $l = \min\{l_1, l_2\}$ on T. Because l_g is not known, conclude $l \leq l_2$ and examine l_2. Clearly $T' \geq 128 \times 192^2 \times T$, hence the lower bound l_2 implied by l_f is *at most*

$$\frac{l_f}{128 \times 192^2}.$$

Because the value of l_f is known, we obtain $l_2 \leq 2^{82}$. The theorem thus establishes that 2^{82} is a bound of type **uL**. The authors aim at "80-bit security" and claim the level is consistent with what the theorem implies. While $2^{80} \leq 2^{82}$, the latter quantity is merely an upper bound on l_2. The original proof does not lead

to a lower bound on l_2. More information on l_g would have to be known and the connection of T and T' would need to be cleaned up.

The quantity 2^{82} counts bit operations. If we want to translate this to the equivalent of hash function computations, divide by the cost of such an evaluation, estimated to be 2^{24} bit operations.[4] The **uL** bound on pre-image resistance then becomes 2^{58} evaluations of the compression function. Existence of a pre-image finding attack with such cost would not contradict the theorem.

Collision resistance There is no proof of collision resistance in [8]. The authors do however sketch an argument in favor of it. Because f is an injection, collisions can only occur in g. Collisions in g are actually easy to find,[5] but in order to lead to collisions in the complete construction, the colliding inputs would have to be in the range of f and that is unlikely. Even if they were, the f is hard to invert. The argument only considers *one particular attack* on collision resistance, hence would only lead to an **U** type bound. Cost of inverting f is equivalent to around 2^{80} evaluations of the compression function, close to the cost of generic collision search, hence the attack is not a serious threat to collision resistance. A proof, however, would require considering *all attacks*, not only a single one.

Because collisions are no harder than pre-images, we might want to make use of the **uL** bound 2^{58} derived there. A **U** bound tied to pre-images would translate to a **U** bound on the cost of collisions. Type **L** bounds need not be preserved. The **uL** bound 2^{58} on pre-images does transfer to the cost of collisions, but carries little useful information. Such a bound needs to be interpreted in the context of the corresponding security proof. In this case it means that Theorem 1 cannot imply a **L** bound on collision resistance that would exceed 2^{58}.

Conclusion The lower bound implied by Theorem 1 is not quantified due to unspecified l_g and looseness of the reduction. The "80-bit" security claimed by the designers is not supported by an **L** bound. We have derived an **uL** bound tied to the proof at 2^{58} evaluations of the function. Improved proofs may be possible.

3.2 FSB

The hash function based on problems in coding theory has a rather long history of provably secure variants (several of them broken) [2, 3, 16, 17]. The most recent variant of FSB was submitted to the NIST SHA-3 competition, but did not advance to Round 2 [15].

Definition The FSB consists of an iterated compression function and a final transformation, that compresses the output further. The compression function

[4] The authors do not comment on how the compression function is to be computed, but their 3 MB memory requirement per evaluation is consistent with our estimate.

[5] This is why a cascade of two systems is used.

is defined as follows: Let H be a $r \times n$ binary matrix, let $s = w \times \lg \frac{n}{w}$ be the input length. Encode the input in a word of length n and weight w, denoted by e. Output the r bits He^T. Denote the compression function by f.

Out of the five FSB variants in [15] we pick FSB_{256} as an example with $n = 2^{21}$, $w = 128$, $r = 1024$ and $s = 1792$. Analogous arguments are possible for the other four variants as well.

Hardness Assumption Security of the compression function is related to two problems from coding theory:

Computational Syndrome Decoding (CSD) Given an $r \times n$ matrix H, an r-bit s and integer $w \leq n$, find $x \in \{0,1\}^n$ such that x has Hamming weight at most w and $Hx^T = s$.

Codeword Finding (CF) Given an $r \times n$ matrix H and integer $w \leq n/2$, find $x \in \{0,1\}^n$ such that x has Hamming weight at most $2w$ and $Hx^T = 0$.

Security Claims The function is proved pre-image resistant reducing to CSD and collision resistant reducing to CF. Both proofs are immediate. There are no explicit lower bounds tied to the assumptions, hence no lower bounds on security are derived. Security is further assessed looking at attacks only. For FSB_{256}, the instances of CSD can be solved in $2^{261.6}$ operations and instances of CF in 2^{153} operations.[6] Both these bounds are due to attacks. Yet the algorithms are not shown to break the actual function, but the more general underlying problems. The reduction goes only one way. A solution to CF or CSD does not imply a collision or a pre-image, respectively. The bounds are therefore of type **uL**.

A more detailed analysis of attacks is performed in [18], estimating cost of two specific algorithms from below, leading to bounds $2^{245.6}$ on CSD and 2^{153} on CF. Again, the problem considered is more general. Extending our notation, their type would be **luL** (i.e. lower bound on a particular **uL** bound). It is suggested these be adopted as **L** bounds [18].

Security is evaluated making use of the **uL** bounds $2^{261.6}$ and 2^{153}. Output of f is 1024 bits long and the security is deep below the trivial bounds, being 2^{1024} and 2^{512}. A final compression is introduced to "fix" this. The 1024 bits are compressed to yield a 256-bit result using another hash function g, instantiated by Whirlpool [5]. The authors remark that "the complexities of the attacks on the FSB compression function ... can thus be transposed directly to the whole hash function and are all above the complexities of generic attacks on the whole FSB ..." Collisions in $g \circ f$ are *no harder* to find than collisions in f. The **uL** bound $2^{261.6}$ thus transfers to $g \circ f$. This is above the trivial **U** bound due to a generic attack. Hence we are left with an **U** bound 2^{256} (that is trivially also an **uL** bound). Such a bound is independent of the hardness assumption.

It might seem that the problem is that the g compresses too much. What if the cost of generic attacks on $g \circ f$ is above the cost of attacks on f? Can the 1024 bits be compressed a little less to maintain some of the provable security?

[6] Counting evaluations of f.

With an output of 320 bits, attacking f might be faster. Still this would only yield an **U** bound, because a collision in $g \circ f$ does not imply a collision in f.

Could lower bounds be preserved? Consider collision resistance. A collision in $g \circ f$ implies a collision in one of the two components. If there were **L** bounds l_f and l_g tied to f and g respectively, the smaller of the two would then be a lower bound on security of the composition. Such proof would be possible if g could be assumed collision resistant in the first place.[7] Although composing the FSB compression function with a provably collision resistant final transformation can preserve the lower bound(s), it would resemble a circular argument where a provably collision resistant function is designed given a provably collision resistant function. This trivial observation only appears in [17] and is left out of the submission to NIST [15].

The authors of FSB consider collision resistance of Whirlpool too strong an assumption [15]. For eventual collisions in Whirlpool to extend to the complete FSB_{256} one needs to invert the FSB primitive. However, saying that collisions for Whirlpool do not easily extend to the complete FSB is an argument from the attack perspective and not a proof. Just as in the case of MQ-HASH earlier, this looks at *particular* attacks and thus does not establish **L** bounds.

Conclusion One claim of the designers in [15] reads as follows:

> The most decisive advantage of FSB is that it comes with a proof of reduction to hard algorithmic problems. An algorithm able to find collisions on FSB or to invert FSB is also able to solve hard problems from coding theory.

No such statement is proved in any of the proposals. The security level claimed by designers is not supported by an **L** bound. This is due to the final compression using Whirlpool. If the step were omitted, **L** bounds on the coding problems would transfer to the compression function. Such bounds were not explicitly provided.

3.3 SWIFFT(X)

SWIFFTX is a SHA-3 proposal based on the simpler primitive SWIFFT [24, 26], not making it to Round 2. It is an example of a generalized knapsack function [27] with security based on hardness of lattice problems.

Definition[8] The SWIFFT compression function takes as input r 64-bit words $x_1, \ldots, x_r$ and outputs 64 elements $z'_0, \ldots, z'_{63} \in \mathbb{Z}_{257}$. The function is indexed by $64r$ fixed elements $a_{1,0}, \ldots, a_{r,63} \in \mathbb{Z}_{257}$ taken to be uniformly random integers modulo 257. Let

$$\text{rev} : \{0, \ldots, 63\} \to \{0, \ldots, 63\}$$

[7] If g is fixed, the assumption is trivially false.

[8] Copied almost verbatim from [1].

be the "bit-reversal" function on 6-bit binary numbers. Output of SWIFFT can be expressed as follows:

$$z'_i = \sum_{j=1}^{r} a_{j,i} \sum_{k=0}^{63} x_{j,\mathrm{rev}(k)} \cdot \omega^{(2i+1)k}$$

where $\omega = 42$, $x_{j,i}$ is the i-th bit of x_j and arithmetic is performed modulo 257. Within SWIFFTX, r equals either 32 or 25.

Hardness Assumption Finding short vectors in lattices isomorphic to ideals of $\mathbb{Z}[\alpha]/\left(\alpha^d + 1\right)$ is hard in the worst case as d increases.[9] The assumption is asymptotic and the **L** bound unquantified. In the light of the results in [19] it was pointed out that for the choice $d = 64$ used in SWIFFT variants as above, the lattice problems are actually easy and the lower bound "insignificant" [10].

Security Claims The SWIFFT function family is proved collision and pre-image resistant [30, 25, 27]. The proof establishes the security properties as the output length increases to infinity.[10] Although asymptotic, the proof does link security of the function for any particular value of d to hardness of a precise lattice problem. We will therefore consider SWIFFT equipped with an **L** bound, yet unknown.

According to the designers in [1]:

> To quantify the exact security of our functions, it is still crucially important to cryptanalyze our specific parameter choices and particular instances of the function.

Instead of finding an **L** bound tied to a proof, the approach chosen is to concentrate on upper bounds due to attacks. While such analysis only establishes bounds of type **U** or **lU**, it does reveal limits of the security proofs provided by the designers of SWIFFT.

The proposal mentions actual attacks on SWIFFT applying the generalized birthday algorithm by Wagner [42]. For $r = 32$, collisions can be found in 2^{106} operations and pre-images in 2^{128} operations. Although the complexities are described as **lU** bounds and estimate the cost of attacks from below, they can be considered good approximations to the actual cost,[11] i.e. bounds of type **U**. Both the bounds are used as such in the proposal to quantify security of the function. Furthermore, the bounds were derived for SWIFFT with $r = 16$ [26]. With $r = 32$ the actual complexities would be lower. While the provable lower bound is not quantified, the two attacks provide **uL** bounds limiting what can be drawn from the proofs available.

[9] Precise statements in [30, 27].
[10] Such arguments have become commonplace in provable security.
[11] Our analysis of the runtime leads to the complexities $2^{106.4}$ and 2^{131}.

For $r = 25$ the authors mention a pre-image finding attack that requires 2^{100} operations. This is an **U** bound, hence also an **uL** bound. Because collisions can be found in at most the same time as pre-images, the same **uL** bound applies to provable collision resistance.

SWIFFTX Existence of the attacks motivated design of the compression function SWIFFTX. It maps 2048 bits to 520 bits combining four calls to SWIFFT with some extra operations in a way that is believed to make the known attacks inefficient. The precise details of the construction can be found in [1]. Care is taken to preserve the provable collision and pre-image resistance. First the input is compressed using three "parallel" instances of SWIFFT with $r = 32$ to yield 1560 bits. A fixed (easily invertible) injection extends this to 1600 bits. Then a single SWIFFT instance with $r = 25$ is applied and 520 bits are output.

The known attacks do not easily extend to SWIFFTX. More precisely, the extended attacks are shown to be more expensive than generic attacks. The construction thus "wipes out" the non-trivial **U** bounds. According to an argument sketched in [1], the construction maintains provable security. A collision in SWIFFTX implies a collision in (at least) one of the four SWIFFT components.[12] Effectively, the *least* of the lower bounds that applied to SWIFFT building blocks is valid for SWIFFTX. We obtain an **uL** bound 2^{100} on provable security for both security properties.

Conclusion The authors of SWIFFTX rely on attacks and claim pre-image resistance 2^{512} and collision resistance 2^{256}. These are not justified by **L** bounds. While the security proofs would lead to **L** bounds, they are not quantified. The improved function SWIFFTX is no more secure than the original SWIFFT primitive in terms of proofs. The **L** bounds implied by the proof provided cannot exceed 2^{100}. As this is an **uL** bound, improved proofs may be possible.

3.4 VSH

The function VSH was introduced in [12] along with a few variants. Some more appeared in [23]. Security of the hash function is linked to hardness of factoring or discrete logarithms.

Definition Let M be an n-bit hard to factor modulus,[13] denote the i-th prime number by p_i. Let k be the largest integer such that $\prod_{i=1}^{k} p_i < M$. Let m be a l-bit message to be hashed, consisting of bits $m_1, \ldots, m_l$ and assume $l < 2^k$. The algorithm runs as follows:

1. Let $x_0 = 1$.
2. Let $\mathcal{L} = \lceil \frac{l}{k} \rceil$. Let $m_i = 0$ for $l < i \leq \mathcal{L}k$.

[12] An analogous argument is possible for pre-image resistance.
[13] Typically a product of two large primes.

3. Let $l = \sum_{i=1}^{k} l_i 2^{i-1}$ with $l_i \in \{0,1\}$ be the binary representation of l and define $m_{\mathcal{L}k+i} = l_i$ for $1 \leq i \leq k$.
4. For $j = 0, 1, \ldots, \mathcal{L}$ in succession compute

$$x_{j+1} = x_j^2 \times \prod_{i=1}^{k} p_i^{m_{j \cdot k+i}} \mod M$$

5. Return $x_{\mathcal{L}}$.

The function iteratively processes blocks of k bits and outputs an n-bit hash. Effectively, it computes a modular k-fold multiexponentiation. The function operates in a variant of the Merkle-Damgård mode processing k bits per iteration. The compression function is not collision resistant, yet the iterated construction is.[14]

Hardness Assumption Given a random M it is hard to find $x \in \mathbb{Z}_M^*$ such that $x^2 \equiv \prod_{i=1}^{k} p_i{}^{e_i} \mod M$ with at least one e_i is odd. The problem is assumed to be k-times easier than the problem of factoring M.

Security Claims The only security property claimed by the designers is collision resistance. Define the function $L'[M]$ to approximate heuristic running time of the Number Field Sieve algorithm factoring the integer M. Assuming this is an **L** bound on hardness of factoring, finding a collision in VSH takes time at least

$$\frac{L'[M]}{k}$$

This **L** bound is used as the basis for security assessment. As an example, collisions in VSH with $n = 1234$ and $k = 152$ are at least as hard to find as it is to factor a 1024 bit (hard to factor) number [12].

No attack is known that would achieve the lower bound. There is an attack and a (non-trivial) **U** bound on security, though. With the knowledge of $\varphi(M)$, collisions can be created easily. Computing $\varphi(M)$ from M is as hard as factoring the modulus. Factorization of M is essentially a trapdoor in the function.

There is an algorithm that finds collisions in VSH factoring the modulus in time approximately $L'[M]$. This is the least **U** bound known. The security of VSH is somewhere between the **L** bound and the **U** bound. So far, no result has appeared that would get the (provable) lower bound closer to the complexity of factoring the modulus M.

Discrete Logarithm Variant of VSH If the modulus chosen to be a prime number of the form $2p+1$ for p a large prime and length of input is limited below $k(n-2)$ bits a VSH-DL compression function is obtained. It is computed in the same way as the basic VSH described above. The function is proved collision

[14] Details in [12].

resistant under a new assumption related to hardness of discrete logarithms in $\mathbb{Z}_M^*$. The assumption is not quantified, hence no lower bound on security of VSH-DL is obtained. Yet an **U** bound is easy to derive, because finding collisions in VSH-DL is no harder than computing discrete logarithms.

Conclusion Collision resistance of basic VSH is supported by an **L** bound and the designers claim precisely the security that is proved. While the DL variant admits a proof, no measure is associated with its hardness assumption. A proof that does not exceed the known **U** bound may be possible.

4 Summary

While several hash function designs claim provable security, only a few actually link security to the complexity associated with the proof.

We gave examples of **uL** bounds that provide partial information on unknown lower bounds. In this way we limit provable security of MQ-HASH and SWIFFT(X) from above. Such arguments are not due to attacks that would actually break the functions. We can view them as *partial attacks* on proofs. We have demonstrated that such incomplete attacks can provide very concrete and useful information on security levels one can prove. Our bounds speak of *particular* proofs, therefore proofs with higher security levels remain possible.

Three of the functions had the structure of compositions. MQ-HASH composed two functions equipped with **L** bounds on pre-image resistance in a way that leads to an unknown **L** bound on security of the composition. FSB composed a function admitting an **L** bound[15] and a function without such a bound. As a result, the proof disappears, while the complexity of attacks is preserved. Finally, the composition within SWIFFTX preserves the proofs and invalidates (some) attacks. While the approaches appear similar, the outcomes differ significantly. Although some general conclusions could be made, bounds in any provable design need to be carefully examined.

We do advocate the use of proofs (i.e. **L** bounds) in design & analysis of hash functions. We hope to have clarified some very basic features of attacks and proofs in hash function security assessment. If there is both an **L** bound and an **U** bound, the former should be pronounced the security level. We believe the function cannot be considered provably secure otherwise. If more security is claimed, this is based on attacks rather than on proofs, rendering the proofs somewhat useless. If it is believed that security is greater than what the proofs suggest, attempts should be made to raise the **L** bound.

Our results should not be viewed as recommendations against or in favor of any of the functions but rather as suggestions where to look for improvements. A tighter reduction within MQ-HASH and quantified hardness of g might well lead to an **L** bound that exceeds 2^{58}. The FSB may as well have a decent **L** bound if the final transformation is omitted. More conditions on the last step g

[15] The assumption is not explicitly stated in the proposals.

might even allow an **L** bound to be established for the complete construction. Quantified hardness assumptions behind SWIFFT and VSH-DL would also lead to precise **L** bounds.

Acknowledgements The author would like to thank Arjen Lenstra, Martijn Stam, Kenny Paterson and the anonymous reviewers for useful comments on the text.

References

1. Yuriy Arbitman, Gil Dogon, Vadim Lyubashevsky, Daniele Micciancio, Chris Peikert, and Alon Rosen. SWIFFTX: A Proposal for the SHA-3 Standard. Submission to NIST, 2008.
2. Daniel Augot, Matthieu Finiasz, and Nicolas Sendrier. A fast provably secure cryptographic hash function. Cryptology ePrint Archive, Report 2003/230, 2003. `http://eprint.iacr.org/`.
3. Daniel Augot, Matthieu Finiasz, and Nicolas Sendrier. A family of fast syndrome based cryptographic hash functions. In Ed Dawson and Serge Vaudenay, editors, *Mycrypt*, volume 3715 of *Lecture Notes in Computer Science*, pages 64–83. Springer, 2005.
4. Jean-Philippe Aumasson and Willi Meier. Analysis of multivariate hash functions. In Kil-Hyun Nam and Gwangsoo Rhee, editors, *ICISC*, volume 4817 of *Lecture Notes in Computer Science*, pages 309–323. Springer, 2007.
5. Paulo S. L. M. Barreto and Vincent Rijmen. The Whirlpool hashing function. Submitted to NESSIE, September 2000. Revised May 2003. Available: `http://www.larc.usp.br/~pbarreto/WhirlpoolPage.html`.
6. Rana Barua and Tanja Lange, editors. *Progress in Cryptology - INDOCRYPT 2006, 7th International Conference on Cryptology in India, Kolkata, India, December 11-13, 2006, Proceedings*, volume 4329 of *Lecture Notes in Computer Science*. Springer, 2006.
7. Daniel J. Bernstein, Tanja Lange, Ruben Niederhagen, Christiane Peters, and Peter Schwabe. Implementing Wagner's generalized birthday attack against the SHA-3 round-1 candidate FSB. Cryptology ePrint Archive, Report 2009/292, 2009. `http://eprint.iacr.org/`.
8. Olivier Billet, Matthew J. B. Robshaw, and Thomas Peyrin. On building hash functions from multivariate quadratic equations. In Josef Pieprzyk, Hossein Ghodosi, and Ed Dawson, editors, *ACISP*, volume 4586 of *Lecture Notes in Computer Science*, pages 82–95. Springer, 2007.
9. John Black, Phillip Rogaway, and Thomas Shrimpton. Black-box analysis of the block-cipher-based hash-function constructions from PGV. In Yung [46], pages 320–335.
10. Johannes Buchmann and Richard Lindner. Secure parameters for SWIFFT. In Bimal K. Roy and Nicolas Sendrier, editors, *INDOCRYPT*, volume 5922 of *Lecture Notes in Computer Science*, pages 1–17. Springer, 2009.
11. Denis Xavier Charles, Kristin E. Lauter, and Eyal Z. Goren. Cryptographic hash functions from expander graphs. *J. Cryptology*, 22(1):93–113, 2009.
12. Scott Contini, Arjen K. Lenstra, and Ron Steinfeld. VSH, an efficient and provable collision-resistant hash function. In Serge Vaudenay, editor, *EUROCRYPT*, volume 4004 of *Lecture Notes in Computer Science*, pages 165–182. Springer, 2006.

13. Jean-Sebastien Coron and Antoine Joux. Cryptanalysis of a provably secure cryptographic hash function. Cryptology ePrint Archive, Report 2004/013, 2004.
14. Ivan Damgård. A design principle for hash functions. In Gilles Brassard, editor, *CRYPTO*, volume 435 of *Lecture Notes in Computer Science*, pages 416–427. Springer, 1989.
15. Daniel Augot and Matthieu Finiasz and Philippe Gaborit and Stéphane Manuel and Nicolas Sendrier. SHA-3 proposal: FSB. Submission to NIST, 2008.
16. M. Finiasz, P. Gaborit, and N. Sendrier. Improved fast syndrome based cryptographic hash functions. *ECRYPT Hash Function Workshop 2007*, 2007.
17. Matthieu Finiasz. Syndrome based collision resistant hashing. In J. Buchmann and J. Ding, editors, *PQCrypto 2008*, volume 5299 of *Lecture Notes in Computer Science*, pages 137–147. Springer, 2008.
18. Matthieu Finiasz and Nicolas Sendrier. Security bounds for the design of code-based cryptosystems. In Mitsuru Matsui, editor, *ASIACRYPT*, volume 5912 of *Lecture Notes in Computer Science*, pages 88–105. Springer, 2009.
19. Nicolas Gama and Phong Q. Nguyen. Predicting lattice reduction. In Smart [37], pages 31–51.
20. M. Grassl, I. Ilić, S. Magliveras, and R. Steinwandt. Cryptanalysis of the Tillich–Zémor Hash Function. *Journal of Cryptology*, 2010.
21. Neal Koblitz and Alfred Menezes. Another Look at "Provable Security". II. In Barua and Lange [6], pages 148–175.
22. Neal Koblitz and Alfred Menezes. Another look at "provable security". *J. Cryptology*, 20(1):3–37, 2007.
23. Arjen K. Lenstra, Dan Page, and Martijn Stam. Discrete logarithm variants of VSH. In Nguyen [28], pages 229–242.
24. V. Lyubashevsky, D. Micciancio, C. Peikert, and A. Rosen. Provably Secure FFT Hashing. *2nd NIST Cryptographic Hash Function Workshop*, 2006.
25. Vadim Lyubashevsky and Daniele Micciancio. Generalized compact knapsacks are collision resistant. In Michele Bugliesi, Bart Preneel, Vladimiro Sassone, and Ingo Wegener, editors, *ICALP (2)*, volume 4052 of *Lecture Notes in Computer Science*, pages 144–155. Springer, 2006.
26. Vadim Lyubashevsky, Daniele Micciancio, Chris Peikert, and Alon Rosen. SWIFFT: A modest proposal for FFT hashing. In Kaisa Nyberg, editor, *FSE*, volume 5086 of *Lecture Notes in Computer Science*, pages 54–72. Springer, 2008.
27. Daniele Micciancio. Generalized compact knapsacks, cyclic lattices, and efficient one-way functions. *Computational Complexity*, 16(4):365–411, 2007.
28. Phong Q. Nguyen, editor. *Progress in Cryptology - VIETCRYPT 2006, First International Conference on Cryptology in Vietnam, Hanoi, Vietnam, September 25-28, 2006, Revised Selected Papers*, volume 4341 of *Lecture Notes in Computer Science*. Springer, 2006.
29. National Institute of Standards and Technology. Announcing request for candidate algorithm nominations for a new cryptographic hash algorithm (SHA3) family. *Federal Register*, 72(212):62212–62220, nov 2007.
30. Chris Peikert and Alon Rosen. Efficient collision-resistant hashing from worst-case assumptions on cyclic lattices. In Shai Halevi and Tal Rabin, editors, *TCC*, volume 3876 of *Lecture Notes in Computer Science*, pages 145–166. Springer, 2006.
31. C. Petit, K. Lauter, and J.J. Quisquater. Cayley Hashes: A Class of Efficient Graph-based Hash Functions. Preprint, 2007.
32. Christophe Petit, Kristin Lauter, and Jean-Jacques Quisquater. Full cryptanalysis of lps and morgenstern hash functions. In Rafail Ostrovsky, Roberto De Prisco, and

Ivan Visconti, editors, *SCN*, volume 5229 of *Lecture Notes in Computer Science*, pages 263–277. Springer, 2008.
33. Christophe Petit, Jean-Jacques Quisquater, Jean-Pierre Tillich, and Gilles Zémor. Hard and easy components of collision search in the Zémor-Tillich hash function: New attacks and reduced variants with equivalent security. In Marc Fischlin, editor, *CT-RSA*, volume 5473 of *Lecture Notes in Computer Science*, pages 182–194. Springer, 2009.
34. Bart Preneel, René Govaerts, and Joos Vandewalle. Hash functions based on block ciphers: A synthetic approach. In Douglas R. Stinson, editor, *CRYPTO*, volume 773 of *Lecture Notes in Computer Science*, pages 368–378. Springer, 1993.
35. Phillip Rogaway. Formalizing human ignorance. In Nguyen [28], pages 211–228.
36. Markku-Juhani Olavi Saarinen. Security of VSH in the real world. In Barua and Lange [6], pages 95–103.
37. Nigel P. Smart, editor. *Advances in Cryptology - EUROCRYPT 2008, 27th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Istanbul, Turkey, April 13-17, 2008. Proceedings*, volume 4965 of *Lecture Notes in Computer Science*. Springer, 2008.
38. Martijn Stam. Blockcipher-based hashing revisited. In Orr Dunkelman, editor, *FSE*, volume 5665 of *Lecture Notes in Computer Science*, pages 67–83. Springer, 2009.
39. Marc Stevens, Arjen K. Lenstra, and Benne de Weger. Chosen-Prefix Collisions for MD5 and Colliding X.509 Certificates for Different Identities. In Moni Naor, editor, *EUROCRYPT*, volume 4515 of *Lecture Notes in Computer Science*, pages 1–22. Springer, 2007.
40. Jean-Pierre Tillich and Gilles Zémor. Hashing with SL_2. In Yvo Desmedt, editor, *CRYPTO*, volume 839 of *Lecture Notes in Computer Science*, pages 40–49. Springer, 1994.
41. Jean-Pierre Tillich and Gilles Zémor. Collisions for the LPS expander graph hash function. In Smart [37], pages 254–269.
42. David Wagner. A generalized birthday problem. In Yung [46], pages 288–303.
43. Xiaoyun Wang, Yiqun Lisa Yin, and Hongbo Yu. Finding collisions in the full SHA-1. In Victor Shoup, editor, *CRYPTO*, volume 3621 of *Lecture Notes in Computer Science*, pages 17–36. Springer, 2005.
44. Xiaoyun Wang and Hongbo Yu. How to break MD5 and other hash functions. In Ronald Cramer, editor, *EUROCRYPT*, volume 3494 of *Lecture Notes in Computer Science*, pages 19–35. Springer, 2005.
45. Bo-Yin Yang, Chia-Hsin Owen Chen, Daniel J. Bernstein, and Jiun-Ming Chen. Analysis of QUAD. In Alex Biryukov, editor, *FSE*, volume 4593 of *Lecture Notes in Computer Science*, pages 290–308. Springer, 2007.
46. Moti Yung, editor. *Advances in Cryptology - CRYPTO 2002, 22nd Annual International Cryptology Conference, Santa Barbara, California, USA, August 18-22, 2002, Proceedings*, volume 2442 of *Lecture Notes in Computer Science*. Springer, 2002.