Applications of SKREM-like symmetric key ciphers

Mircea-Adrian Digulescu[1,2]

[1]Individual Researcher, Worldwide
[2]Formerly: Department of Computer Science, Faculty of Mathematics and Computer Science, University of Bucharest, Romania

Abstract

In a prior paper we introduced a new symmetric key encryption scheme called Short Key Random Encryption Machine (SKREM), for which we claimed excellent security guarantees. In this paper we present and briefly discuss how some other cryptographic applications besides plain text encryption can benefit from the same security guarantees. We task ourselves with and succeed in showing how Secure Coin Flipping, Cryptographic Hashing, Zero-Leaked-Knowledge Authentication and Authorization and a Digital Signature scheme which can be employed on a block-chain, can all be achieved using SKREM-like ciphers, benefiting from their security guarantees. We also briefly recap SKREM-like ciphers and the core traits which make them so secure. The realizations of the above applications are novel because they do not involve public key cryptography. Furthermore, the security of SKREM-like ciphers is not based on hardness of some algebraic operations, thus not opening them up to specific quantum computing attacks.

Keywords: Symmetric Key Encryption, Provable Security, One Time Pad, Zero Knowledge, Cryptographic Commit Protocol, Secure Coin Flipping, Authentication, Authorization, Cryptographic Hash, Digital Signature, Chaos Machine

1 Introduction

So far, most encryption schemes able to serve Secure Coin Flipping, Zero-Knowledge Authentication and Digital Signatures, have relied on public key cryptography, which in turn relies on the hardness of prime factorization or some algebraic operation in general. Prime Factorization, in turn, has been shown to be vulnerable to attacks by a quantum computer (see [1]). In [2] we introduced a novel symmetric key encryption scheme, which does not rely on hardness of algebraic operations for its security guarantees.

1.1 Prior work

The SKREM cipher was introduced in [2]. To the best of our knowledge no such system has been proposed before or since. Nevertheless, we, the author, strongly suspect that non-public research, by researchers such as Marius Zimand (see [3] and [4]) and Leonid Levin (see [5]) might include something similar to SKREM. Nevertheless, to the best of our knowledge, such research, if it exists, is still not public. Chaos Theory and namely Chaos Machines, best described by Czyzewski in [6], can be employed as a black-box subroutine in SKREM. The symmetric key cipher introduced in [2] is conjectured to be *provably* unbreakable. This conjecture is intended to be an educated statement, not a mere shot in the dark and we strongly believe a formal proof exists.

When benchmarking our current approach, we considered popular, well established schemes and methods, such as RSA [7] for public key cryptography, AES (Rijndael) [8]

David C. Wyld et al. (Eds): ITCSE, ICDIPV, NC, CBIoT, CAIML, CRYPIS, ICAIT, NLCA - 2021
pp. 131-147, 2021. CS & IT - CSCP 2021 DOI: 10.5121/csit.2021.110912

for symmetric key encryption, Tang et. al [9] for cryptographic commit protocol, Diffie-Hellman key exchange for secure coin flipping [10], El Gamal [11] and Lamport [12] for digital signatures. Additionally, we considered Elliptic-curve cryptography [13] and the Kerberos authentication and authorization scheme [14].

All of the above, except Lamport and Kerberos which are general purpose methods which can work over symmetric key ciphers, rely on the hardness of algebraic operations such as prime factoring or discreet logarithm which are known [1] and respectively suspected by state actors like the NSA to be vulnerable to quantum computing attacks [15].

The AES cipher itself relies on the hardness of a different algebraic operation, namely the field inverse. While it is not presently know to be vulnerable to quantum attacks, its security claims remain unproven even with regards to a classical computer. Successful practical attacks on reduced versions of AES have been developed [16]. Furthermore, revelations connected with Edward Snowden suggest state actors like the NSA explore using tau statistic to break the full AES itself [17] - and these might be successful.

The security of the methods proposed in this paper relies on the security of SKREM. This in turn does not rely on hardness of any algebraic operation. Instead it relies on the properties of sequences sampled in a truly random fashion to be algorithmically random with high probability under a range of transformations. Concretely, it employs a technique called entropy enhancement to increase the length of the secret key by absorbing bits from a random sequence. It does so employing heavily the operation of indirection (memory dereferencing) which is not algebraic: the implied transformation functions involved in SKREM are thus different for each source sequence and expected to be incompressible. Furthermore, it is conjectured that none of them admits any regular structure, except with negligible probability - being sampled essentially at random.

1.2 Overview of this paper

The rest of the paper is organized as follows. In Section 2 we recap the SKREM cipher, its security claims and assumptions thereof, as well as formalize what is meant by SKREM-like ciphers. In Section 3 we discuss how SKREM-like ciphers can be applied to Secure Coin Flipping and Cryptographic Commit Protocol. In Section 4 we describe how such ciphers can be used to compute secure cryptographic hashes. In Section 5 we present Zero-Leaked-Knowledge Authentication and Authorization protocols over public channels, based on SKREM-like ciphers. In Section 6 we discuss how such can be used to generate digital signatures. In Section 7 we draw the conclusion and present avenues for further research. Section 8 offers the brief Vitae of the author. We conclude the paper with Acknowledgments in Section 9.

2 Recap of SKREM-like symmetric key encryption ciphers

Virtually all present day ciphers proceed from the premise that Encryption / Decryption are two functions $ENCR : P \times K \to C$ and $DECR : C \times K \to P$, where P is the universe of plain texts, K is the universe of secret keys and C is the universe of cipher texts. Usually $P = \{0,1\}^n$ and $C = \{0,1\}^n$, while $K = \{0,1\}^k$, for some fixed constant key size k and plain text length n.

SKREM like ciphers [2] introduce the novelty of taking an auxiliary input, which is neither plain text, nor key - it is a large master table of truly random bits. By truly random it is understood that they are to be harnessed from nature (eg. from radio/solar noise, mouse movements or quantum computers) from a truly random distribution, are

fully independent and uniformly distributed. This is opposed to them being generated by a classical computer from a short seed. The encryption scheme is thus $ENCR : P \times K \times M \rightarrow C$ and $DECR : C \times K \rightarrow P$.$P$ can be arbitrary $P = \{0,1\}^n$. $K = \{0,1\}^k$ is a short key, which has a minimal length dependent on the processing power of the adversary. Namely, a key of length k is conjectured to offer provably unbreakable symmetric key encryption against an adversary with a compute power of $O(2^k)$ - namely one which has enough resources to run this many operations on a classical computer within the relevant attack timeframe. The master table $M = \{0,1\}^m$ is a special input which is required to consist of a truly random bit sequence. Its length $m = O(n)$ is linear in the size of the plain text; however the constant for the original SKREM cipher is around 100,000. Further refinements and simplifications can be made to better practicalize the cipher, with their security being the topic of active research.

The idea behind SKREM is to obtain a random permutation of m elements, based on the secret key k and the master table M. Since the key is short, it does not have enough entropy to generate a permutation which can be argued to be secure. Instead, the very contents of the grand master table M is used to gradually enhance the length of the secret key, in a manner that is uniform and fully unpredictable (and conjectured to be provably so) to any adversary with computing power less than $O(2^k)$. We call this technique entropy enhancement. A subsequence of this permutation is then used to alter the grand master table M, in order to encode the plain text.

In [2] we presented a full SKREM-like cipher and a simplified version - SKREMS. The method is however general and many variations and adaptations can be developed. The following describes the pseudo code of SKREM-like ciphers in general.

Algorithm 1. ***SKREM-like encryption****. Short key random encryption machine. Input:* $k \in K$, $m \in M$ *and* $p \in P$. *Output:* $c \in C$.

1: Split the bits of k into u groups and seed $u \geq 2$ CSPRNGs, $S_1 ... S_u$, with a seed of length $z > log(M)$. It is recommended that $u \cdot z = k$.
2: **while** u is still too small or not enough rounds have completed **do**
3: Double u, creating u new, inactive - yet unseeded CSPRNGs. Intertwine them alternatively with the original ones in the sequence S: $s_1, s_{u+1}, s_2, s_{u+2}, ...$
4: **while** there exists a CSPRNG which does not have all the seed-bits filled out **do**
5: Sample values v_1 and v_2 from two successive, active CSPRNGs in S, starting at index i. Then increment i by 2.
6: Use values v_1 and v_2 to determine two uniformly distributed, random locations, l_1 and l_2 in M, from those which have not yet been visited.
7: Use the values at $M[l_1]$ and $M[l_2]$, if they are different, to generate a new uniformly distributed random bit b.
8: Distribute the random bit b to some index of a new CSPRNGs which does not yet have its seed completed. Distribute it such that each old CSPRNGs contributes an equal number of bits to each new CSPRNG's seed.
9: If $M[l_1] = M[l_2]$ then use then use v_1 and v_2 to permute some $O(1)$ elements/chunks of M, as well as of S.
10: When a priory inactive new CSPRNG has gathered enough bits for a full seed, mark it as active and start using it.
11: Mark the positions l_1 and l_2 of M as visited.
12: **end while**
13: Consider the concatenated seeds of all CSPRNGs in the sequence $S_1 ... S_u$ as a single $u \cdot z$ bit number x. Use a bijective (preferably one way) function $f : \{0,1\}^{u \cdot z} \rightarrow \{0,1\}^{u \cdot z}$, and set $x \leftarrow f(x)$. Then consider back x as a sequence of u CSPRNGs

with seeds z-bits long.

14: **end while**

15: When u is large enough (eg. $u \geq 2 \cdot P$) and enough iterations have passed, repeat steps 5-11 for the n bits of the input, with the difference that the bit produced at step 8 is not used for new CSPRNGs, but instead is a bit of the plain text. When encrypting, $M[l_1]$ and $M[l_2]$ can be exchanged conveniently, if needed, to represent the desired bit.

Complexity. The original SKREM and SKREMS presented in [2] had a space and time complexity linear in the size of the plain text - $O(n)$ for an n-bit plaintext. However, the constant factors were rather large - on the order of 100,000. This can cause incurring a significant running time penalty when the grand master table M is too big to fit in RAM memory. Further sophistication in the implementation of the CSPRNGs (eg. using a Chaos Machine with many iterations per PULL operation), in that of shuffling the locations in M and S, and in the execution of the function f in step 13, could add to the running time of some SKREM-like ciphers - however most often just in terms of constant factors, not of asymptotic complexity. SKREM-like ciphers also require as input a series of truly randomly generated bits, on the order of O(n) - again, with a significant constant factor, on the order of 2-300,000.

The method proceeds to incrementally add new CSPRNGs, by seeding them with the added entropy it gets from the master table M. The order in which the locations of this master table are visited is then conjectured to be a cryptographically secure, random permutation of length m. This conjecture relies on the following assumptions.

Assumption 1. *An arbitrary subset of sufficient length of a truly random sequence m, is itself truly random, except with negligible probability.*

Discussion. This says that if you remove some bits from a truly random sequence m, what remains is still truly random, except in astronomically improbable cases (eg. you chose a subset which consists precisely of the true bits).

Assumption 2. *An arbitrary permutation of truly random sequence m, is itself truly random, except with negligible probability.*

Discussion. This says that if you choose to shuffle the elements of a truly random sequence, the result is still truly random - except in astronomically improbable cases (eg. you chose a permutation which sorts m).

The above assumptions are quite natural and probably admit proofs, given the fact that truly random sequences are expected to be incompressible and thus algorithmically random, except with negligible probability: if the above were not true, constructive martingales could potentially be found which succeed on such sequences.

From the above assumptions, the following conjecture is derived.

Conjecture 1. *A member of the set of permutations producible for a given key k by the Algorithm 1 is a random and unpredictable permutation, except with negligible probability.*

Discussion. The conjecture is quite natural and is suspected to admit a formal proof. This stems from the two facts. Firstly, any suffix of the permutation is fully independent from its prefix - the prefix is derived from the secret key k, and a sampling of the positions of M which are then never reused in the suffix. While any such prefix is uniquely determined by the secret key k (for a fixed M), by induction such a prefix is random and unpredictable. Secondly, the bits which are used to extend the prefix are uniformly distributed by construction, and based on the sequence determined by some sampling from the outstanding portion of M . By Assumptions 1 and 2 these are themselves truly

random except with negligible probability - this means the constructed sequence itself is random and unpredictable, except with negligible probability: otherwise a successful martingale could be constructed.

Finally, based on Conjecture 1, the security of SKREM-like ciphers can be argued to be at least 2^{2z-1}: an attacker will not be able to gain any meaningful insight from deducing (or guessing) some locations which together encode some bit of plain text - except so as in to guess the bits which were used to generate the seed for the CSPRNGs which generated them. Since only a small number of bits are used from each prior-round CSPRNG for each new CSPRNG (preferably only 1 - which is always possible in later rounds), in order to speculate any known plain-text advantage, the attacker will need to guess the full seed of some CSPRNG - z bits. Then he will need to validate if it is plausible, by guessing at least another $z-1$ bits from the resulting new CSPRNG in order to determine if 1 bit of the plain-text is producible by such a priory guessed seed. In practice he will probably need another $\approx z$ bits to gain any meaningful level of confidence for successful guesses.

Attacks are further complicated by the unpredictable mixing of bits in steps 1, 9 and 13 which are intended to mask the correlation between prior-round bits and current round ones, while maintaining statistical properties.

The choice of Cryptographically Secure Pseudo-Random Number Generators (CSPRNGs) can be Chaos Machines [6] which present the advantage that a small variation in source input results in a large variation of output, which is claimed unpredictable except by knowing the seed. Alternatively they can be any CSPRNG such as those based on PRNGs with proven statistical properties [18]. The outputs of CSPRNGs need to be normalized in steps 6 and 9 before they can be used. The original paper on SKREM [2] included a method for converting any distribution to uniform binary and also one for sampling a number in an arbitrary interval $0...p-1$ based on some uniformly distributed bits. It entails using a larger number of bits than strictly required by the desired output.

For the transformation in step 15, alternatively to a large CSPRNG, an algebraic transformation such as the modular or field inverse, followed by appropriate distribution conversion can be used instead. Other various can be sought up, for example using a time-expensive classical one-way function - such as the one described in [5] (which is one-way contingent on the existence of such functions).

Ultimately, the security of SKREM-like ciphers rests in the fact they rely essentially on indirection operation over some large chunk of randomly initialized memory. So long as the chunk is algorithmically random (which being truly random entails except with negligible probability) and the algorithm does not introduce patterns, the result of these operations cannot be predicted at all. The hardness of reversing indirection relays not on difficulty of solving some mathematical problem such as those of modular algebra, but instead on the unpredictability of truly random sequences - their property that no constructive martingale can succeed over them. While for a regular cipher, the transformation function is fixed, for a SKREM-like cipher there are 2^M such functions, one for each original master table, the vast majority of them being secure - not even theoretically breakable, as conjectured.

3 Cryptographic Commit Protocol and Secure Coin Flipping

Consider that Alice and Bob have access to a public master table M, known to be the result of a truly random sampling. While this table is assumed to be public and known to everyone (not only Alice and Bob) without compromising the scheme, it needs to be truly random - it cannot be arbitrary. Now suppose Alice wants to commit to some k-bit value v. Consider the following protocol.

Protocol 1. ***Cryptographic Commit Protocol over SKREM-like ciphers****. Input: Both Alice and Bob have access to a common, immutable, public, truly random master table* M*.*

1: Alice chooses the k-bit value v to which she wants to commit. She also chooses a k-bit truly random secret key s.
2: Alice computes $c \leftarrow DECR_n(M, v \oplus s)$, for a maximal $n < m$, such that M cannot encrypt more than n bits. Then she takes $a \leftarrow c[n/2 - k...n/2 + k]$ to represent the $2k + 1bit$ number comprised of the middle bits of c.
3: Alice sends Bob a.
4: Bob acknowledges receipt of a.
5: Alice sends Bob s.
6: Bob acknowledges receipt of s.
7: Alice and Bob do whatever other interaction, given that Alice has committed to some value v.
8: Alice sends Bob $w = v$.
9: Bob repeats the computations Alice did in step 2, with $v = w$ to get his version of c and a. If Bob's a computed in this round is equal to what he received in step 3 from Alice, he confirms the committal as truthful, otherwise he rejects.

Correctness. Starting with a random master table M, for each key k there is an equal probability of $\frac{1}{2}$ that some arbitrary bit resulting from decryption is set to a particular value. As such, for any t-bit sequence of decrypted output (chosen arbitrarily), there are expected to be $\frac{2^k}{2^t} = 2^{k-t}$ keys which produce it. By sampling $2k + 1$ bits, the probability that there does not exist another value except v which can serve as a secret key to decrypt the same sequence is $(1 - \frac{1}{2^{2k+1}})^{2^{k-1}}$. This tends very quickly to 1 as k tends to infinity. Bob has confidence of at least k-bits security that Alice cannot cheat and that indeed she committed to the value v before step 3 of the algorithm. Furthermore, contingent on the security of SKREM, Bob cannot deduce the secret key v for the "known plain text" a, before Alice reveals it in Step 8.

Complexity. The space/time complexity of the Protocol is dominated by the application of two SKREM decrypt operations, making it linear in the size of the master table M used. In terms of network complexity, only $O(k)$ bits of information are exchanged, making the method robust. The method also requires $O(k)$ truly random bits, on top of the $O(M)$ required by the SKREM-like cipher.

Discussion. Note that if Alice can secretly manipulate the random grand master table M, she could deliberately encrypt a with some key $w \neq v$ chosen by her, and then, in step 8 she would have the choice to send either w or v as her committal value. This is why it is important that the master table M is in fact random, and not chosen or altered by Alice. Conversely, if Bob is allowed to choose M instead, he could theoretically make it that several values decrypt the same sequence (eg. by choosing M to be all zeros). And, as such, he can contest Alice's committal at step 8, by showing a counter example. This is why it is important that the master table is in fact actually random and not chosen by either of the players independently.

The usage of an auxiliary secret key s is because in order for the security claims of SKREM to hold, encryption needs to happen based on a secret key chosen in a truly random fashion, which Alice's committal value v may not be.

Adjusting the above Protocol 1 to support random coin flipping is straightforward.

Protocol 2. ***Secure Coin Flipping over SKREM-like ciphers****. Input: Both Alice and Bob have access to a common, immutable, public, truly random master table* M*.*

1: Alice and Bob perform Protocol 1 above up to step 7, for some arbitrary random value v_A chosen by Alice.
2: Then they again perform the Protocol 1 up to step 7, with Bob committing himself this time to some arbitrary random value v_B chosen by him.
3: They both complete the protocols 1 (in arbitrary order) with the respective confirmations of the truthfulness of the committal. They now both posses both v_A and v_B.
4: They take $v_A \oplus v_B$ to represent their common random value. If this value is over too many bits, they can just ignore a suffix of them.

Correctness. If at least one of them chose randomly, the common random value will be random, since XORing a random number with an arbitrary number maintains the randomness.

Complexity. The protocol essentially consists of two applications Protocol 1 above, thus having the same asymptotic performance characteristics.

Discussion. Note that Alice and Bob could efficiently commit to a large random sequence by using the protocol repeatedly. Note that the values to which either of them commits cannot be over fewer bits than the minimal key size k, required for the security of the SKREM-like cipher used.

4 Cryptographic Hashing

Suppose Alice has some, potentially long, message r, to which she wants to compute a potentially short cryptographic hash. As before, consider a public master table M exists, known to be the result of a truly random sampling, to which Alice has access.

If $|r| = k$, then r could be used directly as a secret key, to obtain a digest by the same method used in Protocol 1 above by Alice to commit to some value, with the secret key s appended to the digest.

If r is longer than k bits it could be used directly in Step 1 of Algorithm 1 to, while keeping z fixed (chosen beforehand), simply start with more CSPRNGs upfront. The stopping condition in Step 2 needs however ensure a reasonable number of rounds happen (at least 3 is recommended) and that, additionally, all original CSPRNGs, at the very least, are used to determine bits at least as many bits so as to exhaust their entropy (namely r in total) - all this before the final round occurs in Step 15. Alternatively, r could be divided into k-bit sequences and the process repeated, as in the case of block ciphers, with the resulting digests appended or combined by some other method. This approach requires that r be truly random however, since SKREM requires truly random input keys.

There is however a more appealing alternative.

Algorithm 2. ***Cryptographically Secure Hashing over SKREM-like ciphers.*** *Input: There exists a public, well known truly random master table M. Alice wants a secure cryptographic hash of message r.*

1: Alice chooses truly randomly a k-bit value s.
2: Alice computes $c \leftarrow DECR_n^r(M, v \oplus s)$, for a maximal $n < m$, such that M cannot encrypt more than n bits. Then she takes $a \leftarrow c[n/2 - r/2 - k...n/2 + r/2 + k]$ to represent the $2k + r + 1bit$ number comprised of the middle bits of c. The function $DECR_n^r$ is the same as that of Algorithm 1, except that, in Step 8, before an obtained bit b is used, it is XORed with the next unused bit from r, until all such are exhausted.
3: The secure hash is <h,s>.

Correctness. There are $2^r \times 2^k$ combinations of potential messages times secret keys. The probability there does not exist another pair to generate the same fixed $r + 2k + 1$ bits is $(1 - \frac{1}{2^{r+2k+1}})^{2^{r+k}-1}$. This again ensures security of at least k bits, for sufficiently large r and k. The underlying implicit assumption is that the probability for a <message, key> pair to produce a certain sequence of bits over M is uniform. This is natural given Conjecture 1.

Complexity. The space/time complexity of the method is given by the application of a single SKREM decrypt operation, making it linear in the size of the master table M used. The method also requires $O(k)$ truly random bits, on top of the $O(M)$ required for the master table M.

Discussion. Note that the size of the message r cannot be arbitrary large, but needs to be small enough for all of its bits to be used by the altered $DECR_n^r$ function. Choosing M so that $|r| \approx n$, where n is defined as in step 2 of Algorithm 2, should prove adequate for the SKREM-like ciphers proposed in [2].

Note that by using Algorithm 2, the obtained bits of the hash are essentially fully independent from those of r. In fact, such a hash would be different for different original master tables M.

In case the hash digest length of $r+2k+1$ is too great, it could be reduced (preferably to k) by using a classical cryptographic hash function, such as Whirlpool [19], SHA-2 [20] or SHA-3 [21] as the underlying hash of a Merkel Tree [22], whose root, together with the secret key s give the digest. The added advantage in this situation is that the cryptographic hash function is computed over an $r+2k+1$ bit sequence which is chosen truly randomly, and whose bits are not related to that of the original message r. The original bits of r are only used, together with those of s and with a discarded portion of M, to select which random sequence of $r + 2k + 1$ bits from the outstanding portion of M is chosen. By Assumptions 1 and 2 and Conjecture 1, this will be a truly random sequence, except with negligible probability - even in cases when r is very regular, such as all zeros.

Alternatively, the hash length could be reduced to some arbitrary length x by taking fewer bits around the middle of c in step 2 of Algorithm 2, with the drawback of increased risk of collisions.

Yet a better option is to use a Merkel Tree [22] but with this very Algorithm 2, with a reduced digest of size k taken with the idea above, instead of a classical cryptographic hash function such as SHA or Whirlpool. We recommend this third option in practice.

Note that the master table M is required to verify a hash. It is assumed to be immutable, public and available over the lifecycle of the generated hashes.

5 Zero Knowledge Authentication and Authorization

Suppose Alice and Bob are secret agents of the same agency, who have never met before and don't know each other, but were given a shared secret key k by their common HQ beforehand. Consider that Alice and Bob have access to a public master table M, known to be the result of a truly random sampling.

Now suppose Alice and Bob want to establish to one-another that they are part of the same organization. However, they want to achieve this without giving any knowledge about the shared secret key k, neither to their counterparty, nor to any member of the public who didn't already know it in advance (such as Mallory). They also want to defend against a public, non-insider actor, such as Mallory using the conversation she witnessed to potentially deceive Alice or Bob or someone else in the future into believing they also knew the secret.

Furthermore, suppose Alice and Bob may want to be able to reveal their membership of the secret agency to the other only if the other also does the same - namely Alice wants to reveal to Bob she is a secret agent *iff* Bob reveals the same to her during the execution of the protocol.

Protocol 3. ***Zero Knowledge Secret Agent Authentication over SKREM-like ciphers****. Input: Both Alice and Bob have access to a common, immutable, public, truly random master table M. Alice and Bob belong to the same secret organization, Alpha, and like any such member they were given the same private secret key k. They want to mutually authenticate each other over a public channel, leaking zero knowledge about k.*

1: Alice and Bob both start with an $x = 0$ and a security confidence parameter $p = 1$.
2: During rest of the Protocol, Alice and Bob take turns. Alice starts first.
3: {no_rounds is a sufficiently large, even integer}
4: **for** $i = 0...no\_rounds$ **do**
5: Alice and Bob agree and commit to number of z public, shared, random k-bit values $v_1...v_z$, using the Coin Flipping Protocol 2 of Section 3.
6: For each v_i, in order, Both Alice and Bob compute $c = DECR_n(M, v_i \oplus k)$, for a sufficiently large $n < m$, such that M cannot encrypt more than n bits. They then take $a \leftarrow c[n/2-8...n/2-1]$ and $b = c[n/2...n/2+7]$ to represent two bytes around the middle bits of c. Then they set their respective $x \leftarrow x \oplus b$ and append $a \oplus x$ to a fresh internal sequence s they maintain (sequences s are discarded from round to round).
7: Whose ever turn it is (say Alice) computes a new sequence s_A by taking the internal sequence s and replacing a number of $z - p$ randomly chosen elements in it with random values and then sends it to the counterparty (say Bob).
8: Both Alice and Bob count the number of correct answers in the transmitted sequence, by comparing each to the members of their internal sequence s. Say this number is y.
9: Both Alice and Bob validate that $y \geq p$. Otherwise authentication is rejected, and protocol continues with $p = 0$ immutable.
10: If authentication did not fail above, both Alice and Bob set $p \leftarrow 1 + (y \mod z)$. Also, if $p > tr$ for some specific threshold $tr < z$, authentication is considered successful and the protocol continues with $p = 0$ immutable.
11: The roles of Alice and Bob are switched for the next round.
12: **end for**
13: If the authentication was never declared successful in line 10, then it is declared failed now.

Correctness. If both parties know the secret key k, the value of p will monotonically increase by $+1$ each round, up until $min\{z, no\_rounds\}$. If at least one of them doesn't, by the security of SKREM the best they can do is guess a member of the target sequence. They guess correctly with probability 1/256, meaning, by linearity of expectation, that the expected number of correct values in this case is $\frac{1}{256} \cdot z$. The exact probability that, by chance, the actual number of correct guesses strays too far from this expectation decreases very fast. For illustration, if we chose $z = 10 \cdot 256$ and threshold $tr = z$, the probability of false positives is $\frac{1}{256}^{10 \cdot 256} = 2^{-20480}$ which is negligible. We recommend choosing $z \geq 256$, for there to be positive expected value for the number of correct hits by chance. The exact value of tr depends on the desired maximal probability of deceit, resulting in a false positive authentication. Finding a good value of tr is left as an exercise for further research. It can be computed algorithmically for a given false positive probability. We expect it be less than 62 for $z = 10 \cdot 256$. This value should also be used as no_rounds.

Complexity. The running time complexity of the method is given by z applications of a SKREM decrypt operation for each round - of which there are z in total maximum, making it $O(z^2 * M)$. Since the master table M is not changed, the space complexity is just $O(M)$. The method also requires $O(z^2)$ truly random bits, on top of the $O(M)$ required for the master table M. Since z is expected to be a small constant, these are asymptotically excellent characteristics.

Discussion. Note that Alice and Bob reveal each others' knowledge of the shared secret with increasing probability, in tandem. If Bob wants to withhold his knowledge of the shared secret, the authentication will fail for both sides. He will at most learn that Alice had the ability to guess $+1$ more than he revealed about his own ability to guess. If the expected number of correct guesses by chance is large enough (say more than 10 above expected value), this added information should be insufficient for a confident appraisal. Therefore, in order to learn about Alice's membership to the organization he must reveal his own as well.

An outside observer will not be able to as much as decide which from the values he sees were not chosen at random in step 7, much less be able to determine a full sequence of such. Even if they were to learn the exact sequence, by security of SKREM they would be unable to determine the encryption key $k \oplus v_i$ for such a "known plain text". Also, since the values v_i are chosen at random via a cryptographically secure coin flipping protocol, they will not occur again in future instances of the protocol's execution - not even individually, much less so as the entire sequence for all the rounds -, except with negligible probability.

Protocol 3 offers a way for Alice and Bob to (almost) simultaneously prove to each other, over a public channel, that they have knowledge of a shared secret, without leaking any information about the secret itself, not even theoretically, to the rest of the channel participants, who did not already know it. Do note however that any other participant to the channel who did know the secret can determine if authentication of Alice and Bob was successful by listening in. In order to prevent this, they should communicate over a secured, bilateral channel, at least when committing to the v_i sequences.

Note that a knowledgeable third party listening in on Alice's and Bob's chatter cannot determine for sure that they are actually secret agents: they could both NOT be and simply replay a conversation overheard priory between actual secret agents over the same master table M. In order for authentication to be genuine, it must occur over a "never before seen" truly random master table M, or occur interactively.

Protocol 3 opens the door way to a host of interesting related applications. Suppose now that Alice and Bob have determined they are both secret agents, they want to find out who has the highest rank, so they know who gives and who takes the orders.

Suppose the ranking hierarchy of the secret organization Alpha is a linear chain, with the members of each rank r being given, upon promotion, not only the secret key k_r corresponding to their own rank, but also another one, k_{r+1} corresponding to the immediately superior rank, whom they must obey. Also, the secret agency does not want to reveal to all agents how many ranks there are exactly in the organization. Also, when authorizing themselves, agents do not want to reveal their exact rank, but only that they are of superior rank to the counterparty, if that is so.

This can be achieved by following the following protocol.

Protocol 4. ***Zero Knowledge Secret Agent Authorization over SKREM-like ciphers****. Input: Both Alice and Bob have access to a common, immutable, public, truly random master table M. Alice and Bob belong to the same secret organization, Alpha. Alice's rank is r and she knows keys $k_1 ... k_{r+1}$ and Bob's rank is q and he knows secret keys $k_1 ... k_{q+1}$. They want to mutually authenticate each other over a public channel, leaking zero knowledge and also to determine who of them is of higher rank.*

1: Let $last \leftarrow 0$, $left \leftarrow 0$ and $right \leftarrow MAX$, with MAX a value known to be greater than the number of ranks existing in the secret agency.
2: **while** $left \leq right$ **do**
3: Alice and Bob set $mid \leftarrow (left + right)/2$.
4: Alice and Bob try to authenticate each other over rank mid, with shared secret key k_{mid}. If one of them is of rank lower than k_{mid-1}, he or she answers randomly to the challenges of the rank, thus causing authentication to fail.
5: If authentication succeeds above, they both set $last \leftarrow mid$ and $left \leftarrow mid + 1$.
6: If authentication fails in step 4 above, they both set $right \leftarrow mid - 1$.
7: **end while**
8: To determine who is in charge both Alice and Bob compare $last$ to their own rank. If it is greater, than the other party is their superior, otherwise they are.
9: In order to obscure the actual number of rounds the protocol took, step 4 is performed an additional number of times, in order to bring the total to $log(MAX)$ - the results of these extra rounds are ignored.

Correctness. Essentially, the parties binary search to find the highest rank about which they both know. The person who knows of a higher rank than that is clearly the superior. The party of inferior rank is only able to confirm that the counterparty is her superior, but cannot establish his exact rank.

Complexity. The method consists of a number of applications of a Protocol 3 above, which is logarithmic in the number of ranks in the Alpha secret organization. Thus, the running time is $O(log(MAX) * z^2 * M)$, the space complexity remains unchanged at $O(M)$ since the same master table is used for all applications. There is a need for an additional $O(log(MAX) * z^2)$ extra truly randomly generated bits, except the $O(M)$ used by the master table.

Discussion. The protocol can be performed not only bilaterally but also with regard to a third party, say Claire. Claire may be an automated weapons system - such as a Poseidon or Minuteman strategic nuclear article, and Alice and Bob two competing secret agents who want to give conflicting orders to Claire. By having both Alice and Bob perform the protocol with her (not with each other), Claire can decide to whom to listen. This also gives no indication to the other party as to the actual rank of the counterparty, or any information that could help it in the future to pass a similar authorization protocol.

There is one added bonus for using Protocols 3 and 4: **stenography**. Since only a small fraction of the actual elements of the sequences exchanged are required to have fixed values, the rest can be used to **"piggyback a secondary transmission on the same carrier wave"**. This allows for example a double agent Alice to perform an authentication protocol for secret agency Alpha with Bob, while at the same doing an authentication protocol for secret agency Omega with the same Bob. This way, any member of agency Alpha listening in on the conversation will believe that Alice and Bob simply authenticated each other for agency Alpha, when instead they might have also established that they both belong to agency Omega also. Furthermore, in case Bob does not belong to agency Omega (or does not wish to reveal his belonging), Alice's attempt for mutual authentication for Omega will remain a secret: neither Bob nor other members of Omega on the public channel will be able to ascertain that Alice attempted such.

Clearly such a protocol would have been useful to the conspirators of the Lodge of Perfect Equality in the time of the 1789 French Revolution, to authenticate one-another while seemingly simply authenticating that they are simple members of the Freemasonry, not also part of some conspirator group within it.

When the organization Alpha is a well-known public organization, such as an Internet market place, authenticating against it may seem natural to the observers. This gives the

pretext to piggyback a secondary message (for all intents and purposes seemingly random) as part of the authentication protocol. This can be used not only to authenticate within Omega, but also to send and receive encrypted information which is indistinguishable from random (eg. encrypted with a SKREM-like cipher). Such information will be short over a single transmission, but not of trivial length. It can include specific encrypted orders, or can form portions of a longer message transmitted over several sessions.

Protocols 3 and 4 could be adapted to function with other symmetric key ciphers. However, their current formulation presents the advantages of the security guarantees of SKREM-like ciphers in general. Existing alternatives rely on hardness of some algebraic problem such as discreet logarithm (like [23]) - many of which are clearly vulnerably to quantum computing attacks -. Or they employ the trapdoor approach over NP-complete problems (like [24]) - which approach, in many cases, has been shown to be breakable in practice.

Unlike Kerberos [14], the presented protocols do not require a trusted third party (authentication server) and can be enacted bilaterally. On the other hand, Kerberos itself can be extended to use Protocols 3 and 4 to authenticate and respectively authorize a new client when issuing him a TGT key. Just as well, a SKREM-like cipher can be used as the symmetric key cipher for Kerberos.

6 Digital Signatures

Suppose Alice wants to be able to digitally sign some arbitrary message r, by producing a digital signature $sig(r)$, such that no one else is able to produce such a signature except her and that all members of public are able to verify and be confident that it is indeed her who signed it.

Alice can proceed as follows. Firstly, she generates a secret, truly random grand master table M. Then she has several options. One of them is for her to employ the Lamport digital signature scheme [12], and use the cryptographic hashing Algorithm 2 from Section 4 to compute the cryptographic hashes involved. Then she can publish M unmodified, together with her public key. This has the advantage that she can use any master table M, including some potentially naturally occurring ones. However, the public key can be used only once.

There is, however an even better possibility.

Algorithm 3. ***Digital Signatures over SKREM-like ciphers****. Input: Alice posses some secret, truly random master table M. She wants to be able to digitally sign messages of length r bytes.*

1: Alice computes some k-bit truly random values $v_{i,j,k}$ and secret keys $k_{i,j,k}$, for $i = 1..n$, $j = 1..r$ and $k = 0..255$, with n being the number of messages she plans to sign using the public key.
2: Alice computes the following plain texts $m_{i,j,k} = \langle v_{i,j,k}, j, k, i, n, hash(\langle v_{i,j,k}, j, k, i, n\rangle)\rangle$, for $i = 1..n$, $j = 1..r$ and $k = 0..255$. The hash function can be any cryptographically secure hash fuction with a short digest, such as Algorithm 2 from Section 4 over M.
3: Alice sets $C \leftarrow ENCR(m_{i,j,k}, k_{i,j,k}, M)$, encrypting all $256 \cdot n \cdot r$ messages in the same cipher text, using the methodology described in [2] which leverages Universal Perfect Hashing. If the function used above is indeed Algorithm 2 over M, the locations touched by any of the hashing operations must also be protected, the same way as when encrypting.
4: Alice reveals C as her public key.

5: When Alice wants to sign the i-th public message, consisting of bytes $r_1, r_2, ..., r_r$, she publishes $\langle k_{i,j,r[j]} \rangle$ for all $j = 1..r$.

6: Someone wishing to verify her signature checks that $DECR(C, v_j) = \langle x, j, r[j], i, n, hash(\langle x, j, r[j], i, n \rangle) \rangle$ for the r values $v_1 ... v_r$ published by Alice as signature. This involves checking the hash of the deciphered output and checking that the second and third values in the tuple are indeed j and $r[j]$ as expected. The verification succeeds *iff* the check holds for all $j = 1..r$.

Correctness. Similar to Lamport signature, for an attacker to be able to forge Alice's signature over even a 1 byte long message b, he would need to find at least some key k such that $DECR(C, k) = m_{i,1,b}$ for some arbitrary i. But the probability of a naturally occurring such $m_{i,1,b}$ outside those purposefully minted by Alice is less than the probability of hash collisions for the hashing function. This in turn is negligible if Algorithm 2 from Section 4 is used: less than 2^{-k}, which should exceed the compute power available to an adversary.

Complexity. As it is presented, the Algorithm involves encrypting $O(n * r)$ plaintexts within the same master table M. The size of each plain-text is dominated by that of the digest produced by the hash function in step 2. Using Algorithm 2 from Section 4, that is on the order of $O(k)$. This bounds the running time complexity to $O(n * r * k)$. Computing each hash, using the raw Algorithm 2 from Section 4 involves a full $O(M)$ SKREM decrypt operation for each, bringing the total to $O(n * r * k * M)$, which is rather large - even if incurred only one time, during the generation of the public part of the signature. A practical implementation, will most likely adjust the hash algorithm to involve only up to $O(k)$ steps, thus making signature generation take overall $O(n * r * k)$ time. Signing is linear in the message size, namely $O(r)$. Verifying a signature takes one SKREM decrypt operation and one hash operation for each byte of message. The latter takes $O(M)$ using the raw Algorithm 2 and just $O(k)$ using the suggested practicalization. The overall complexity of verification can thus reach $O(r*k)$. Note that besides the master table M, Alice needs another $256 * n * r * k$ bits for the secret keys $k_{i,j,k}$ and a roughly similar amount for the SKREM encrypt operations.

Discussion. Note that the total size of all encrypted messages is $O(n \cdot r)$. This can grow large for large n or large r. In case the message to be signed is very large, she can instead sign a cryptographic hash digest of it. Note that in fact the maximum values for j can be made to vary from one i to another. This means she could "set aside" some of the signatures from the n allotted, to sign longer messages.

In case Alice runs out of signatures, she could resort to signing a special (potentially long) message containing the digest of a fresh public key C_{new}, consisting of a suitably altered new truly random master table M. This should be reserved only for the last $i = n$ of the signatures she uses, and could be understood by the public as such. Note that while she can sign a digest for a new public key which is longer than the usual messages she signs, that digest can never be as long as her original public key. So, if the number of available signatures and security guarantees for the new public key are to be maintained, only a digest of such can be signed. Computing this digest with Algorithm 2 of Section 4 over her original public key C, with the idea to use a Merkel Tree over the same algorithm to shorten the digest should prove enough for ensuring security. No maleficent party can create a fresh public key even after seeing C_{new}, thanks to the security guarantees of the Algorithm 2.

While the signature scheme described is not based on public key cryptography, it could nevertheless be used on a block-chain [25] to sign transactions. A public digest of the private key could be published beforehand as part of the incoming transaction (similar to how hashes of public keys current work on the BitCoin block-chain) and the full public

key revealed and used to sign an outgoing transaction. Since transactions on the BitCoin block chain, do not generally reuse public keys, the number of messages that would need to be signable by a single public key is as low as $n = 1$. In order to support spending inputs of large-valued transactions we can take $n = 2$ or $n = 3$. Thus, the total size of a public key would by reasonably small (linear in the size of the digest). The security of such an approach relays of course on the security of the hashing digest function. The one described in Section 4 for example, was argued to be secure and not reliant on hardness of prime factoring or of any algebraic operation in general.

The described scheme can also be used in secure authorization: A client with a certain level of clearance (perhaps proven via Protocol 4 of Section 5) can formulate certain types of requests, including a user ID and timestamp with them. An Authorization Server could receive such request from an authenticated client, authorize them and then send back a digital signature of such, signed with its own private key (the Authorization Server's). The client could then directly send such requests to a Serving Server which could then simply check the validity of the digital signature. Note that the Serving Server could already have the Authorization Server's public key C, so the two do not need to be connected live at the moment of serving.

This allows for scenarios like the following: Say there are number of underwater mines (say Poseidon strategic articles) deep behind enemy lines, or within neutral waters. They are all programmed with the same public key C of the authorization server. Then, a dully authorized party wants to issue an order (eg. attack / self-destroy / do not attack and remain dormant for at another week / no operation) to one (including its ID in the request) or all of them. The authorization server could be located deep below ground in a bunker in the Urals and be totally disconnected from any of the mines. The authorized party can compose the order message, get it signed by the secure authorization server and then transmit it to the mine (eg. via satellite link, marine mammals, submarines or surface ships). The mine will execute the order *iff* it is authorized and without the need to communicate with the authorization server. Finally, what is interesting is that even if the enemy were to capture one mine, and then even if they were able to reverse engineer it and discover the Authorization Server's public key C - and even if they were able to intercept a message addressed to that particular mine -, they would still be unable to fool any of the other mines into executing any orders.

The ideas in this paper can also be combined to secure a top secret algorithm which is embedded in some forward-deployed hardware and which is activated only on command: It can be encrypted with SKREM, and its encryption key be sent over as part of the authorized request. Since SKREM supports multiple plain-texts over the same cipher note how this could allow export-versions of military hardware to be turned into strategic-versions, by the simple issuing of an authorized command.

7 Conclusions And Further Research

The ideas in this paper have demonstrated the usefulness of SKREM-like ciphers over a range of applications.

We also recaped SKREM-like ciphers and argued the extraordinary promise they present: provably unbreakable symmetric key encryption, with short secret keys. While the fact their security is provably unbreakable remains a conjecture presently, arguments have been presented why this is very likely to be provable shortly. SKREM-like ciphers present other advantages also: they are not block ciphers, but instead operate with the entire plain text, however long it may be. This means there are no vulnerabilities associated with the choice of some chaining method, like XTS, needed to turn constant-sized block ciphers

like AES into stream ciphers. Furthermore, SKREM is not based on algebric operations. At its core, it leverages the operation of indirection - thus even a small, 1 bit change in its source can result in a fully independent, unpredictable result. This makes SKREM able to counter quantum computing attacks which are known and respectively suspected or feared for some other symmetric key and public key ciphers.

For all these benefits, SKREM only requires that it be provided with a rather large - yet linear - number of truly randomly generated bits. Furthermore, both encryption and decryption are asymtotically optimal.

Neverthless, SKREM as introduced in [2] is slow. The constat factors of about 100,000 are still too large for a large range of practical applications. An avenue of immediate further research is to further perfect SKREM-like ciphers, reducing the constant factors to something more manageble, without compromising the security guarantees.

Additionally, proving the Conjectures and/or Assumptions on which the security guarantees of SKREM-like ciphers rely is expected to be a major milestone in cryptography since this essentially solves symmetric key encryption, even in the post-quantum computing era. In doing that, a natural first step is to fully implement a SKREM-like cipher and analyze its operation statistically. It is conjectured that it produces output indistinguishible from random, except with negligible probablity - so the cipher texts it produces should pass all randomness tests with flying colors.

Further beyond, expanding the applications of SKREM-like ciphers into related areas such as Public Key Cryptography and Turing-complete functional encryption is desired and could be the subject of further research as well.

One aspect which was taken rather for granted with SKREM is the generation of truly random numbers. The approach proposed involved harnessing such randomness from hardware sources - these can include natural phenomena, such as inbound solar radiation or FM noise, a quantum computer or a human user moving the mouse repeatedly. Alternatively, they could be sampled from a sequence produced by a third party trusted for the use case (such as random.org perhaps). For truly sensitive applications however, further research is required into how acceptable master tables can be generated, verified and distributed securely. We expect a lot of practical attacks on SKREM to focus on the low quality and improper reuse of master tables.

Finally, both to allow further analysis of SKREM-like ciphers, and "ideal candidates" therefrom, an area of direct further development is to provide a library implementing SKREM, and the methods based on SKREM above. This can then be integrated into existing private or open source packages, like VeraCrypt [26], making the promise of provably unbreakable security with short secret keys - and its applications - widely available and generally adopted.

We conclude this paper here.

In "good old fashioned" tradition of publications in cryptography we offer 20$ to the first 14 people who show an attack on SKREM. Furthermore, we offer an additional 28$ to the first 9 people who invalidate Conjecture 1 or Assumptions 1 and 2.

8 Acknowledgments

The ideas behind SKREM-like ciphers was inspired by the author's exposure to chaos theory as a child, while visting the Paris science museum Cité des Sciences et de l'Industrie. Specific details were inspired by examining the field and some elementary results pertaining to Kolmogorov extractors by Zimand, M. (see [3] and [4]). The formalization of Chaos machines in [6] is both used as a potential subroutine in SKREM and served as a catalyst for faster organization of the present results in written form.

The secret agent authentication and authorization schemes are based on ideas the author had in the summer of 2015, for a "protocol for mutual recognition of synonymouas". A synonymous is a human person, of female gender, semantically equivalent to love.

Warm thanks for their existence and patient understanding to the few beautiful persons who inspired the author to strive to make the world a better place for them, and furthermore served as an inspiration for some of the constants used in the algorithms. This paper would never have existed without them.

9 Authors

Mircea Digulescu is an independent computer science researcher, software engineer, entrepreneur, military and intelligence enthusiast and amateur writer.

He has a PhD (ABD) at the University of Bucharest, Department of Computer Science of Faculty of Mathematics and Computer Science, from which he also obtained both a Bachelors and a Master's degree in Computer Science. He gained recognition by being awarded medals and prizes at international contests and Olympiads in Computer Science, including 1 Bronze Medal at CEOI and 2 prizes at ACM SEERC. He is a Div1 coder on Codeforces.com. He also has over 15 years of experience in the software industry, creating systems that currently run in production and scale to billions of transactions.

His research interests include Cryptography, Game Theory, Complexity Theory as well as Algorithms and Data Structures.

References

[1] Shor, P.W., "Algorithms for quantum computation: discrete logarithms and factoring", *Proceedings 35th Annual Symposium on Foundations of Computer Science.*, 1994, 124-134.

[2] *Digulescu, M., 2019, Hiding Data in Plain Sight: Towards Provably Unbreakable Encryption with Short Secret Key and One-Way Functions. ResearchGate. DOI: 10.13140/RG.2.2.34319.94887.*

[3] Zimand, M., "On the topological size of sets of random strings", *Mathematical Logic Quarterly*, **32(6)** (1986), 81-88.

[4] *Zimand, M., 2009, Extracting the Kolmogorov complexity of strings and sequences from sources with limited independence. arXiv. arXiv:0902.2141.*

[5] Levin, L.A., "The tale of one-way functions", *Problems of Information Transmission*, **39(1)** (2003), 92-103.

[6] Czyzewski, M.A., "Chaos Machine: Different Approach to the Application and Significance of Numbers", *IACR Cryptol. ePrint Arch.*, 2016, 468.

[7] Rivest, R.L., Shamir, A. and Adleman, L., "A method for obtaining digital signatures and public-key cryptosystems", *Communications of the ACM*, **21(2)** (1978), 120-126.

[8] Daemen, J. and Rijmen, V., "The block cipher Rijndael", *In International Conference on Smart Card Research and Advanced Applications*, 1998, 277-284.

[9] Tang, C., Pei, D., Liu, Z. and He, Y., "Non-Interactive and Information-Theoretic Secure Publicly Verifiable Secret Sharing", *IACR Cryptol. ePrint Arch*, 2004, 201.

[10] Diffie, W. and Hellman, M., "New directions in cryptography", *IEEE transactions on Information Theory*, **22(6)** (1976), 644-654.

[11] Taher ElGamal, "A Public-Key Cryptosystem and a Signature Scheme Based on Discrete Logarithms", *IEEE Transactions on Information Theory*, **31 (4)** (1985), 469-472.

[12] Lamport Leslie, "Constructing Digital Signatures from a One Way Function", *SRI International (CSL-98)*, 1979.

[13] Kobliz, N., "Elliptic curve cryptography", *Math. Comput.*, **48** (1987), 203-209.

[14] Steiner, J.G., Neuman, B.C. and Schiller, J.I., "Kerberos: An Authentication Service for Open Network Systems", *Usenix Winter*, 1988, 191-202.

[15] *National Security Agency, US, 2015, Elliptic curve cryptography. Commercial National Security Algorithm Suite.*

[16] Biryukov, A., Dunkelman, O., Keller, N., Khovratovich, D. and Shamir, A., "Key recovery attacks of practical complexity on AES-256 variants with up to 10 rounds", *Annual International Conference on the Theory and Applications of Cryptographic Techniques*, 2010, 299-319.

[17] "Inside the NSA's War on Internet Security", 2014.

[18] Ryabko B., and Zhuravlev V., "Pseudo-random number generators with proven statistical properties", *The 9th Workshop on current trends in Cryptology*, 2020.

[19] Barreto, P.S.L.M. and Rijmen, V., "The Whirlpool hashing function", *First open NESSIE Workshop, Leuven, Belgium*, 2000.

[20] Penard, W. and van Werkhoven, T., "On the secure hash algorithm family", *National Security Agency. Tech. Rep.*, 2008.

[21] Dworkin, M.J., "SHA-3 standard: Permutation-based hash and extendable-output functions", *Federal Inf. Process. STDS*, 2015.

[22] Merkle, R.C., "A digital signature based on a conventional encryption function", *Conference on the theory and application of cryptographic techniques*, 1987, 369-378.

[23] Chaum, D., Evertse, J.H. and Van De Graaf, J., "An improved protocol for demonstrating possession of discrete logarithms and some generalizations jWorkshop on the Theory and Application of of Cryptographic Techniques", 1987, 127-141.

[24] Blum, M., "How to prove a theorem so no one else can claim it", *Proceedings of the International Congress of Mathematicians*, **1** (1986), 2.

[25] Damle, A., Bangera, M., Tripathi, S. and Meena, M., "Blockchain Technology: An Overview", *SAMRIDDHI: A Journal of Physical Sciences, Engineering and Technology*, **12(SUP 1)** (2020), 243-247.

[26] Bursać, M., Vulović, R. and Milosavljević, M., "Comparative Analysis of the Open Source Tools Intended for Data Encryption", *International Conference on Information Technology and Development of Education – ITRO 2017*, 2017.