Customization Modes for the Harris MK-3 Authenticated Encryption Algorithm

Peter Bajorski[*], Alan Kaminsky[†], Michael Kurdziel[‡], Marcin Łukowiak[§], and Stanisław Radziszowski[¶]
[*]School of Mathematical Sciences, Rochester Institute of Technology, pxbeqa@rit.edu
[†]Department of Computer Science, Rochester Institute of Technology, ark@cs.rit.edu
[‡]Harris Corporation, MKurdzie@harris.com
[§]Department of Computer Engineering, Rochester Institute of Technology, mxleec@rit.edu
[¶]Department of Computer Science, Rochester Institute of Technology, spr@cs.rit.edu

***Abstract*—MK-3 is a new proprietary authenticated encryption algorithm based on the duplex sponge construction. To provide security autonomy capability, such that different users can have sovereign variants of the encryption algorithm, MK-3 is designed to be customizable. Two levels of customization are supported, Factory Customization and Field Customization. Customization is done by modifying functions and function parameters in the algorithm to yield differing cipher functions while preserving the algorithm's security. This paper describes the MK-3 algorithm's customization options and discusses results of testing designed to verify security autonomy among the customized variants.**

I. Introduction

The MK-3 authenticated encryption algorithm [1][2] uses the duplex sponge construction [3]. As such, the algorithm's design centers on a bijective function that maps a 512-bit input state to a 512-bit output state. The bijective function consists of several rounds: 10 rounds for a 128-bit key, 16 rounds for a 256-bit key. Each round consists of a substitution layer with 16×16-bit S-boxes, a bit permutation layer, a mixer layer, and a round constant addition layer.

The MK-3 algorithm is intended to be customizable. Each customized version must yield a different encryption algorithm that is not interoperable with any other version, while still being as secure as the original algorithm analyzed by Kelly [1].

This paper's novel contributions are threefold. First, whereas the original paper [1] described just the basic MK-3 algorithm, this paper describes modifications to the basic algorithm's design to implement customized versions of the algorithm. Second, this paper analyzes the cryptographic security of customized versions of the algorithm. Third, this paper analyzes whether customized versions of the algorithm are noninteroperable with each other and with the original version.

The MK-3 algorithm supports two levels of customization: Factory Customization and Field Customization. The Factory Customization capability allows different customized encryption algorithms to be provided to various customers. The user cannot affect this level of customization. Rather, settings are specified for a customized version, after analysis to ensure that the customized version remains secure. These settings are stored in firmware and are loaded into the encryption circuitry at power-on. Factory Customization is implemented in any or all of the bijective function layers.

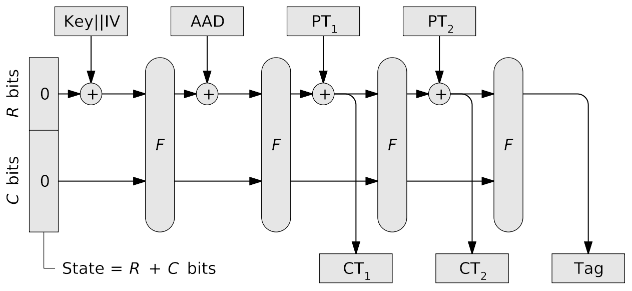

Fig. 1. MK-3 duplex sponge construction

The Field Customization capability allows the user to customize the encryption algorithm further after power-on. These settings are stored in a 128-bit register that can be changed at any time during operation (except in the middle of a message). All possible register values must yield different, fully secure customized algorithms. Field Customization is implemented in the bijective function's mixer layer.

This paper is organized as follows. Section II describes the MK-3 authenticated encryption algorithm. Section III describes the requirements for military security autonomy among different users of a cryptographic algorithm. Section IV describes how MK-3's Factory Customization and Field Customization are implemented and explains why customized versions are still secure. Section V reports results of experiments to determine whether different customized MK-3 versions are in fact noninteroperable. Section VI offers concluding remarks.

II. MK-3 Authenticated Encryption Algorithm

MK-3 is a new authenticated encryption algorithm that meets government and military security requirements. It is based on a simplified version of the duplex sponge construction [3]. Refer to Fig. 1. Padding and field separation are applied at higher levels in the system. For this reason, it is sufficient to specify only the MK-3 state and the MK-3 bijective function F.

The MK-3 state size is 512 bits, consisting of a 128-bit rate portion R and a 384-bit capacity portion C. With this rate and capacity, MK-3 can support key sizes of 128 bits and 256 bits. These are the NIST recommended symmetric key sizes [4].

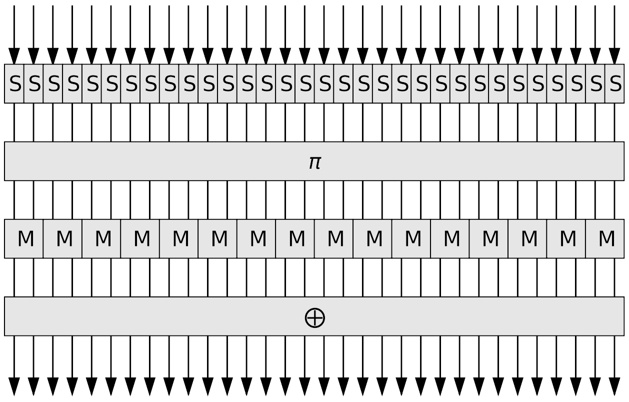

Fig. 2. MK-3 round function; each line is one 16-bit word

The MK-3 bijective function consists of a number of iterated rounds: 10 rounds for a 128-bit key, 16 rounds for a 256-bit key. Fig. 2 shows the round function, comprising four steps: the Substitution Step (S), the Bitwise Permutation Step (π), the Mix Step (M), and the Add Round Constant Step ($\oplus$). The 512-bit state is partitioned into thirty-two 16-bit words, and operations are applied to these words.

The first operation in the MK-3 round function is the Substitution Step (S). The substitution step is a bricklayer permutation that uses 32 identical, bijective 16×16-bit S-boxes. This step is the main source of confusion within the cipher. It is the only nonlinear step, as is typical with most substitution-based symmetric key algorithms [5].

The second operation is the Bitwise Permutation Step (π). Bitwise permutations are easily implementable in hardware via a simple rerouting of wires. Compared to a permutation on the words of the state, a bitwise permutation intuitively provides much better diffusion. The bitwise permutation step is the main source of long-range diffusion (i.e., across the entire state) in the algorithm.

The third operation is the Mix Step (M). The purpose of the mix step is to provide local diffusion (i.e., across two words) and to increase the linear and differential branch numbers of a round from two to three. MK-3 uses a mixer based on multiplication by a 2×2 matrix in GF(2^{16}) modulo the irreducible polynomial $p(x) = x^{16} + x^5 + x^3 + x^2 + 1$. The mixer takes a vector of two input words A and B and multiplies the vector by an invertible matrix, producing a vector of output words A' and B':

$$\begin{pmatrix} A' \\ B' \end{pmatrix} = \begin{pmatrix} 1 & x \\ x & x+1 \end{pmatrix} \times \begin{pmatrix} A \\ B \end{pmatrix}$$

The final operation is the Add Round Constant Step ($\oplus$). A constant 512-bit value is added to the state using bitwise exclusive-or. To disrupt symmetry and prevent slide attacks, a different round constant is added in each round.

III. Military Security Autonomy

Sovereign cryptography refers to the capability that allows cryptography users to install their own cryptographic algorithm into a product after delivery and without U.S. Government or radio vendor involvement. This provides the customer with the capability of "Security Autonomy." Security Autonomy is defined as the ability to manage the security posture of an information system in a way that is independent of any third party. This not only includes independent operational management of the system, but also independent design and maintenance of the security elements of the system.

The most significant challenge to providing sovereign cryptographic capability is policy related. The U.S. Department of State export policy regulates the international distribution of military-grade cryptographic technologies and specifically limits sovereign cryptography approaches in military communications equipment. Development and economic feasibility, deployment logistics, system security verification and system reliability testing present additional challenges.

For military applications, a Security Autonomy capability can also be provided by supplying users with their own customized versions of a proprietary cryptographic algorithm. The MK-3 cryptographic algorithm can be customized in various ways. This capability is provided with two types of customization, namely Factory Customization and Field Customization.

Factory Customization is substantial in that the structure and major components of the cryptographic algorithm are modified. One disadvantage of Factory Customization is that human error can cause degradation of the cryptographic system's security. MK-3 employs an encryption/decryption algorithm structure that can be customized under well understood design constraints. Proper Factory Customization and verification allows a custom algorithm variant to be provided with maximum security.

Users might also want to restrict knowledge of their own variant of a cryptographic algorithm. To this end, the MK-3 design allows the algorithm to be customized in the field. Field Customization allows users to make changes to the cryptographic algorithm via a tool after the device is provided to them. All possible parameters that can be input into the system via the tool to provide the Field Customization are equally valid, and none degrade the cryptographic strength of the algorithm. In addition, parameters for this mode of customization are known only to the user.

IV. MK-3 Customization Implementation

Customized versions of the MK-3 algorithm are implemented by altering the round function in specific ways designed to preserve the algorithm's security while ensuring that different customized versions of the algorithm are not interoperable. Factory customizations affect any or all steps of the round function. Field customizations affect the Mix Step.

A. Factory Customization

The MK-3 round function is designed to be customizable. The user cannot change the customized features once they are installed at the factory. The parameters and functions are designed to preserve the algorithm's security; consequently, each potential group of parameters and functions must be analyzed to ensure they meet security requirements.

1) Substitution Step Factory Customization: The MK-3 S-box computes the following function of the 16-bit input α:

$$S(\alpha) = A \cdot \alpha^{-1} + b$$

The input is treated as an element of GF(2^{16})/$f(x)$, where $f(x)$ is a degree-16 irreducible polynomial; the inverse α^{-1} is computed; the inverse is treated as a 16-bit vector and an affine transform is applied, yielding the 16-bit output. The affine transform is defined by a 16×16-bit invertible matrix A and a 16-bit vector b. The MK-3 paper [1] specified an S-box with a particular irreducible polynomial $f(x)$, matrix A, and vector b.

The MK-3 security analysis in [1] was based on these characteristics of the S-box:

- No fixed points, where $S(\alpha) = \alpha$.
- No opposite fixed points, where $S(\alpha)$ = bitwise complement of α.
- Maximum differential probability $\leq 2^{-14}$.
- Maximum linear bias $\leq 2^{-8}$.

However, the security analysis did not assume any particular values for the irreducible polynomial $f(x)$, the matrix A, or the vector b. Therefore, the Substitution Step can be customized, without affecting the algorithm's security, by changing any or all of $f(x)$, A, and b, yielding a different S-box mapping.

A tool set is used to create a customized S-box. Given a degree-16 irreducible polynomial $f(x)$, the tool generates a random invertible matrix A and a random vector b; analyzes the resulting S-box to verify that it meets the preceding security requirements; and repeats if necessary with different A and b until the S-box is suitable.

2) Bitwise Permutation Step Factory Customization: The Bitwise Permutation Step in [1] rearranges the 512 bits of the state via this formula:

$$\pi(x) = 31x + 15 \pmod{512}$$

where x is the input bit position ($0 \leq x \leq 511$) and $\pi(x)$ is the output bit position ($0 \leq \pi(x) \leq 511$).

The MK-3 security analysis in [1] was based on these characteristics of the bitwise permutation:

- Each output bit of a given S-box goes to a different mixer's input bit.
- The bitwise permutation has no fixed points, where $\pi(x) = x$.
- The order of each bit position is greater than the number of rounds in the bijective function.

As the bijective function goes through multiple rounds, each bit position in the input state is repeatedly permuted by applying the above formula. Repeatedly applying the permutation formula to a bit position—that is, computing $\pi(\pi(\ldots(\pi(x))))$—eventually yields the same bit position as the original x. The "order" of a bit position is the number of repetitions needed for a bit to return to its original position. A bit that returns to the same position during the rounds of the bijective function is a weakness that an attacker might be able to exploit. (For example, an attack on the PRESENT block cipher was successful due in part to its round function's bitwise permutation, which has fixed points and small orders [6].)

Any bitwise permutation formula that meets the preceding security requirements may be used without degrading the security of MK-3. Kelly [1] identified 384 suitable permutation formulas. The factory customizes the Bitwise Permutation Step by choosing one of these formulas.

3) Mix Step Factory Customization: As will be seen, Field Customization of the Mix Step uses a fixed 256-element lookup table. Each table element gives the coefficients of a different degree-16 irreducible polynomial. The table's contents are established during Factory Customization. The factory customizes the table by choosing 256 different polynomials from among the 4080 possible degree-16 irreducible polynomials.

4) Add Round Constant Step Factory Customization: The Add Round Constant Step exclusive-ors a 512-bit constant into the state. Each round uses a different round constant. Kelly [1] specified particular round constants.

The security of MK-3 depends on these characteristics of the round constants:

- Every round constant is different.
- There is no structure in the round constants.

However, the security analysis does not depend on the particular values of the round constants. Therefore, the Add Round Constant Step can be customized, without affecting the algorithm's security, by choosing different round constants that meet the preceding requirements.

A recommended method for creating round constants is to apply a cryptographic hash function, such as SHA-512 or SHA-3-512, to compute the digests of successive counter values starting from an arbitrary value, and to use the digests as the round constants.

B. Field Customization

MK-3 includes a 128-bit Field Customization Register (FCR) that performs additional customization beyond the factory installed function and parameter settings. The FCR's contents may be changed at any time during operation. All of the possible FCR values yield fully secure, yet noninteroperable customized versions of the MK-3 algorithm. The user can therefore pick any FCR value without needing to analyze the security of the resulting version.

As previously stated, each mixer in the Mix Step treats its inputs as field elements in GF(2^{16}), that is, polynomials. The mixer's outputs A' and B' are computed from its inputs A and B by these formulas:

$$\begin{aligned} A' &= A + xB \\ B' &= xA + xB + B \end{aligned}$$

The mixer's output computations involve field multiplications. A GF(2^{16}) field multiplication is the polynomial product of the field elements modulo an irreducible degree-16 polynomial

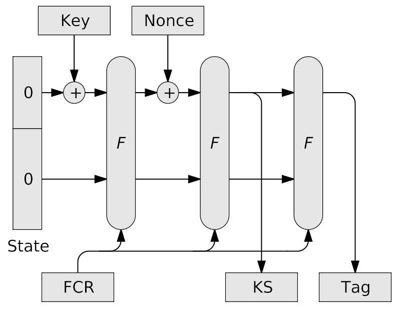

Fig. 3. MK-3-based authenticated stream cipher, 128-bit key

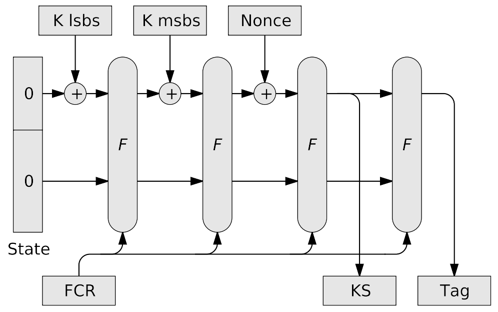

Fig. 4. MK-3-based authenticated stream cipher, 256-bit key

$p(x)$. Kelly [1] specified the mixer's irreducible polynomial as $p(x) = x^{16} + x^5 + x^3 + x^2 + 1$.

However, the security analysis does not depend on the particular choice of the mixers' irreducible polynomial. Choosing a different irreducible polynomial will alter the output values computed by the above formulas, thereby altering the MK-3 round function's mapping, without affecting its security.

The 128-bit FCR is partitioned into sixteen 8-bit sections, one section for each of the sixteen mixers in the round function. Each mixer's circuitry uses the value of the corresponding FCR section as an index into the 256-element irreducible polynomial table described previously to obtain the irreducible polynomial coefficients for that mixer. Note that different mixers can be made to use different irreducible polynomials.

Thus, changing the FCR setting in the field changes the mapping calculated by each mixer, yielding a customized version of the MK-3 algorithm. Because the FCR setting picks each mixer's polynomial out of a factory-specified table of irreducible polynomials, every possible FCR setting is guaranteed to yield a valid mixer mapping.

V. MK-3 Security Autonomy Testing

To evaluate whether Field Customization yields noninteroperable versions of MK-3, we created two software implementations of an MK-3-based authenticated stream cipher, including the FCR. Fig. 3 depicts the first implementation, which inputs a 128-bit key and a 128-bit nonce and outputs a 128-bit keystream (KS) and a 128-bit tag. Fig. 4 depicts the second implementation, which inputs a 256-bit key (split into two 128-bit chunks) and a 128-bit nonce and outputs a 128-bit keystream and a 128-bit tag. The keystream and tag are what would be obtained by encrypting an all-zero plaintext block. (While we used a single keystream block in our tests, multiple keystream blocks might be generated when encrypting an actual message.)

We tested the original MK-3 version from [1] as well as ten factory-customized versions. The customized versions altered the S-box's A matrix and b vector, the Mix Step's table of irreducible polynomials, and the round constants, as described previously.

To test each factory-customized version, we applied various key, nonce, and FCR inputs (described below), we observed the keystream and tag outputs, and we used statistical tests to determine whether the outputs showed random behavior or nonrandom behavior. Of course, a secure cipher should generate random outputs.

The statistical tests took the form of odds ratio uniformity tests on a series of outputs (keystream, tag) resulting from a series of differing inputs (key, nonce, FCR). The odds ratio uniformity test uses the methodology of Bayesian model selection [7]. The odds ratio uniformity test calculates the logarithm of the posterior odds ratio of two hypotheses: H_1, that the series of output values obeys a discrete uniform distribution, and H_2, that the series of output values does not obey a discrete uniform distribution. If the log odds ratio is positive, then H_1's probability is greater than H_2's probability, indicating that the output series is random. Vice versa, if the log odds ratio is negative, then H_2's probability is greater than H_1's probability, indicating that the output series is nonrandom. For detailed information about the odds ratio uniformity test, see Appendix B of [8].

We performed four kinds of statistical tests:

- *Avalanche Test (AVAL).* We inputed an all-zero key, an all-zero nonce, and an all-zero FCR, and we observed the keystream and tag outputs. Then we applied a series of inputs, each of which differed from the original input by flipping a single bit in the key, nonce, or FCR, and we observed the series of differences (exclusive-ors) between the resulting outputs and the original outputs. We applied odds ratio uniformity tests to detect nonrandom behavior in the series of differences. These tested whether flipping a single input bit resulted in completely different (random) outputs.
- *Key Varying Difference Test (KVDT).* We inputed an all-zero key, an all-zero nonce, and an all-zero FCR, and we observed the keystream and tag outputs. Then we applied a series of values to the key input while keeping the nonce and FCR the same. The key inputs formed a Gray code sequence, in which successive keys differed in just one bit position. We observed the series of differences (exclusive-ors) between the each output and the previous output. We applied odds ratio uniformity tests to detect nonrandom behavior in the series of differences. These tested whether a small change to the key resulted in completely different (random) outputs.

TABLE I
NUMBER OF NONRANDOM ROUNDS FOR EACH TEST AND MK-3 VERSION, 128-BIT KEY

Version	AVAL	KVDT	NVDT	FVDT
Original	2	1	2	1
Custom 1	2	1	2	3
Custom 2	2	1	2	3
Custom 3	2	1	2	1
Custom 4	2	1	2	3
Custom 5	2	1	2	1
Custom 6	2	1	2	4
Custom 7	2	1	2	3
Custom 8	2	1	2	1
Custom 9	2	1	2	1
Custom 10	2	1	2	4

TABLE II
NUMBER OF NONRANDOM ROUNDS FOR EACH TEST AND MK-3 VERSION, 256-BIT KEY

Version	AVAL	KVDT	NVDT	FVDT
Original	1	1	2	2
Custom 1	1	1	2	2
Custom 2	1	1	2	2
Custom 3	1	1	2	1
Custom 4	1	1	2	2
Custom 5	1	1	2	1
Custom 6	1	1	2	2
Custom 7	1	1	2	2
Custom 8	1	1	2	1
Custom 9	1	1	2	1
Custom 10	1	1	2	2

- *Nonce Varying Difference Test (NVDT).* This was the same as the Key Varying Difference Test, except the nonce was varied rather than the key. These tested whether a small change to the nonce resulted in completely different (random) outputs.
- *FCR Varying Difference Test (FVDT).* This was the same as the Key Varying Difference Test, except the FCR was varied rather than the key. These tested whether a small change to the FCR resulted in completely different (random) outputs.

We tested with the bijective function reduced to one round, two rounds, and so on up to the full number of rounds. With just one round, the outputs were nonrandom (the log odds ratio was negative). As the number of rounds increased, eventually the outputs became random (the log odds ratio was positive). This analysis determines the number of rounds that are required to ensure that the outputs' behavior is random.

Table I gives the results of the AVAL, KVDT, NVDT, and FVDT tests on the original version and the ten customized versions of the MK-3-based stream cipher with a 128-bit key. The table reports the number of nonrandom rounds detected. For example, the AVAL test result says that when the bijective function is reduced to one or two rounds, the test reported nonrandom behavior in the keystream and tag outputs; that is, the values in a series of outputs were not uniformly distributed. With the bijective function reduced to three or more rounds, the test reported random behavior in the outputs. As the full number of rounds is 10 for a 128-bit key and eight rounds exhibited random output behavior, the "randomness margin" for this test is 8/10 or 0.80. The worst-case randomness margin over all the tests is 0.60.

Similarly, Table II gives the results of the AVAL, KVDT, NVDT, and FVDT tests on the original version and the ten customized versions of the MK-3-based stream cipher with a 256-bit key. The full number of rounds is 16 for a 256-bit key; thus, the worst-case randomness margin is 14/16 or 0.88.

Returning to the original question, whether Field Customization yields noninteroperable MK-3 versions: The AVAL and FVDT results show that when the FCR value is changed, the output before the change and the output after the change bear no relationship to each other; more precisely, the difference is a uniformly distributed random variable. Therefore, the ciphertexts and tags produced by one field-customized MK-3 version cannot be successfully decrypted by another field-customized MK-3 version; in other words, the versions are noninteroperable. This holds true both for 128-bit keys and 256-bit keys, provided the bijective function computes the full number of rounds (10 or 16, respectively).

The statistical tests also show that the MK-3 stream cipher generates unrelated keystreams and tags when the key or the nonce is changed. A secure cipher requires this behavior.

VI. CONCLUSION

We have described how changing various parameters in the MK-3 encryption algorithm yields different noninteroperable, yet fully secure, customized versions. Factory Customization provides a sovereign cryptography capability to each MK-3 customer; the factory ensures that the customized version is as secure as the original version. Field Customization allows each individual user to customize the algorithm further without needing to do a security analysis. Statistical tests showed that Field Customization does yield noninteroperable versions.

REFERENCES

[1] M. Kelly. Design and cryptanalysis of a customizable authenticated encryption algorithm. RIT Computer Engineering M.S. thesis, August 2014. http://scholarworks.rit.edu/theses/8325/

[2] M. Kelly, A. Kaminsky, M. Kurdziel, M. Łukowiak, and S. Radziszowski. Customizable sponge-based authenticated encryption using 16-bit S-boxes. In *IEEE Military Communications Conference (MILCOM 2015),* pp. 43–48, 2015.

[3] G. Bertoni, J. Daemen, M. Peeters, and G. Van Assche. Duplexing the sponge: single-pass authenticated encryption and other applications. In *Selected Areas in Cryptography,* pp. 320–337, Springer, 2012.

[4] E. Barker and A. Roginsky. Transitions: recommendation for transitioning the use of cryptographic algorithms and key lengths. *NIST Special Publication 800-131A,* January 2011. http://csrc.nist.gov/publications/nistpubs/800-131A/sp800-131A.pdf

[5] D. Stinson. *Cryptography: Theory and Practice, Third Edition.* Chapman & Hall/CRC, 2006.

[6] J. Nakahara, P. Sepehrdad, B. Zhang, and M. Wang. Linear (hull) and algebraic cryptanalysis of the block cipher PRESENT. In *Cryptology and Network Security,* pp. 58–75, Springer, 2009.

[7] R. Kass and A. Raftery. Bayes factors. *Journal of the American Statistical Association* 90:773–795, 1995.

[8] A. Kaminsky. ElsieFour: a low-tech authenticated encryption algorithm for human-to-human communication. Cryptology ePrint Archive, Report 2017/339. April 12, 2017. http://eprint.iacr.org/2017/339