

PERMUTATION DATA TRANSFORM TO ENHANCE SECURITY

5

10

FIELD

15 This invention pertains to data security, and more particularly to a permutation data transform to enhance security.

BACKGROUND

For thousands of years, man has found it necessary to keep secrets. But for most of history, the art of keeping secrets developed slowly. The Caesar shift cipher, supposedly
20 used by Julius Caesar himself, involved taking a letter and shifting it forward through the alphabet, to hide the message. Thus, "A" became "D", "B" became "E", and so on. Although generally considered a very weak encryption, there were few better encryption algorithms developed until centuries later.

Encryption became a focus of intense research during the two World Wars. Much
25 effort was expended, both in developing codes that the enemy could not break, and in learning how to read the enemy's encrypted mail. Mechanical devices were designed to aid in encryption. One of the most famous of these machines is the German Enigma machine, although Enigma was by no means the only mechanical encryption machine of the era.

The advent of the computer has greatly altered the landscape for the use of
30 encryption. No longer requiring complex machines or hours of manual labor, computers can encrypt and decrypt messages at high speed and for trivial cost. The understanding of the mathematics underlying computers has also introduced new encryption algorithms. The work of Diffie and Hellman led to a way to exchange private keys using exponential arithmetic modulo primes, and relies on the fact that calculating the shared key given the public
35 information is computationally infeasible. And the popular RSA algorithm (named after its

inventors: R. Rivest, A. Shamir, and L. Adleman) relies on the fact that factoring large numbers is also computationally infeasible to decrypt encrypted data. The work of Diffie and Hellman, and the RSA algorithm, can theoretically be cracked, but cracking these algorithms would depend on solving mathematical problems that have yet to be solved. (As an aside, the RSA algorithm was also one of the first public-key cryptosystems, using a different key to decrypt than the key used to encrypt. This made it possible to publicly distribute one key without losing security.)

But no encryption algorithm has an infinite life span. For example, DES (the Data Encryption Standard) was originally released in 1976. The government originally estimated its useful life at 10 years. DES has lasted much longer than the original estimated life span, but because of its relatively short key, DES is considered less than ideal. DES has since been replaced by AES (the Advanced Encryption Standard) as the government standard, but DES remains in widespread use. Various improvements to DES exist, but these improvements cannot make DES secure forever. Eventually, DES will generally be considered insecure.

A need remains for a way to enhance the security of existing encryption algorithms.

SUMMARY

In one embodiment, the invention is a method and apparatus for a permutation data transformation. The data is divided into two segments. Bits in the first segment control the application of permutation functions to bit groups in the second segment. The transformed data includes the first segment and the permuted groups of the second segment.

In a second embodiment, the invention is a method and apparatus for an exponential data transformation. The data is divided into two segments. The second segment is raised to a power of a function of the first segment. A modulus is then applied to the result. The transformed data includes the first segment and the remainder modulo the modulus.

In a third embodiment, the invention is a method and apparatus for performing key derivation from a master key. In one embodiment, a portion of the master key is hashed. Two numbers are derived from another portion of the master key. A universal hash function, using the two numbers, is applied to the result of the hash, from which bits are selected as the derived key.

In a fourth embodiment, a universal hash function, using an encoded counter, is applied to portions of the master key, and the results combined. The combined result is then hashed, from which bits are selected as the derived key.

The foregoing and other features, objects, and advantages of the invention will become more readily apparent from the following detailed description, which proceeds with reference to the accompanying drawings.

5

10

15

20

25

BRIEF DESCRIPTION OF THE DRAWINGS

FIG. 1 shows a general implementation of a secure hash algorithm to generate derivative keys from a master key.

FIG. 2 shows the typical operation of the secure hash algorithm of FIG. 1.

30 FIG. 3 show the typical operation of a universal hash algorithm.

FIG. 4 shows different ways to combine the secure hash algorithm and the universal hash algorithm of FIG. 1 to generate more secure derivative keys, according to an embodiment of the invention.

FIG. 5 shows a server and device capable of performing data transformations, key generation, key wrapping, and data encryption, according to an embodiment of the invention.

FIG. 6 shows a data security device operable to enhance security by using a data transformer in combination with a key wrapper, key deriver, or an encryption function,

5 according to an embodiment of the invention.

FIGs. 7A-7B show a flowchart for using the data security device of FIG. 6, according to an embodiment of the invention.

FIG. 8 shows details of the data transformer of FIGs. 5 and 6, according to an embodiment of the invention.

10 FIG. 9 shows details of the data transformer of FIGs. 5 and 6, according to another embodiment of the invention.

FIGs. 10A-10C show a flowchart for using the data transformer of FIG. 8, according to an embodiment of the invention.

15 FIG. 11 shows a flowchart for using the data transformer of FIG. 9, according to an embodiment of the invention.

FIG. 12 shows details of the key derivation function of FIGs. 5 and 6, according to an embodiment of the invention.

FIG. 13 shows details of the key derivation function of FIGs. 5 and 6, according to another embodiment of the invention.

20 FIG. 14 shows a flowchart for using the key derivation function of FIG. 12, according to an embodiment of the invention.

FIG. 15 shows a flowchart for using the key derivation function of FIG. 13, according to an embodiment of the invention.

25 FIG. 16 shows a flowchart for using a key derivation function in the data security device of FIG. 5, according to an embodiment of the invention.

DETAILED DESCRIPTION

FIG. 1 shows a general implementation of a secure hash algorithm to generate derivative keys from a master key. The general concept is that master key 105 is input to secure hash algorithm 110. An example of a secure hash algorithm is SHA-1 (Secure Hash Algorithm 1). The result is derived key 115-1. Secure hash algorithm 110 can be used multiple times. Depending on the implementation of secure hash algorithm 110, master key 105 can be used repeatedly as input to secure hash algorithm 110 with or without modification. For example, if secure hash algorithm 110 uses a clock to control its output,

30

then master key 105 can be used without modification to generated derived keys 115-2 and 115-3. Otherwise, master key 105 can be combined with a counter in some way to modify master key 105 sufficiently to differentiate derived keys 115-2 and 115-3 from derived key 115-1. If secure hash algorithm 105 is properly implemented, then changing as little as a single bit in master key 105 can result in derived keys 115-2 and 115-3 being completely unrelated to derived key 115-1.

FIG. 2 shows the typical operation of the secure hash algorithm of FIG. 1. As shown, a hash algorithm maps inputs to hash values. In FIG. 2, the hash values vary between 0 and n for some value of n . The output of a hash algorithm can be referred to as baskets; FIG. 2 shows baskets 205, 210, 215, and so on to basket 220.

Unlike a general hash algorithm, which can use any desired mapping to map inputs to baskets, a secure hash algorithm is unpredictable (sometimes also called *collision-free*): knowing that one input produces a particular output does not give any information about how to find another input that would produce the same output. For example, knowing that an input of "5" maps to basket 215 does not aid someone in finding any other input value that would also map to basket 215. In fact, there may be no other inputs that map to basket 215, for some particular hash algorithms. This is what makes secure hash algorithm 110 "secure": that there is no easy way to find another input that maps to a desired output. The only way to find another input that maps to a particular output is by experimenting with different inputs, in the hope of finding another value that maps to the desired output.

The weakness of a secure hash algorithm is that the baskets might not all be mapped to equally. In other words, there might be only one input that is mapped to basket 215, but 100 inputs that map to basket 205. And as mentioned above, some baskets might have no inputs that map to them.

A universal hash algorithm provides the distribution feature that is missing from a secure hash algorithm. As shown in FIG. 3, universal hash algorithm 305 also maps inputs to baskets 310, 315, 320, up to 325. But unlike the secure hash algorithm of FIG. 2, universal hash algorithm 305 distributes its input evenly across the baskets. Thus, basket 310 is mapped to just as often as basket 315, 320, 325, and so on.

The weakness of a universal hash algorithm is that it is typically easy to find other inputs that map to the same basket. For example, consider the universal hash algorithm that maps to 10 baskets, numbered 0 through 9, by selecting the basket that corresponds to the last digit of the input. It is easy to see that this hash algorithm distributes its output evenly across

all baskets. But it is also easy to see how to find another input that maps to the same basket as a given input. For example, 1, 11, 21, 31, etc. all map to basket 315.

Thus, it should be apparent that both secure hash algorithms and universal hash algorithms have advantages and disadvantages. The best solution from the point of view of security would be to somehow combine the advantages of both secure hash algorithms and universal hash algorithms. FIG. 4 shows how the secure hash algorithm of FIGs. 1-2 and the universal hash algorithm of FIG. 3 can be combined to generate more secure derivative keys, according to an embodiment of the invention. In sequence 405, master key 105 is first passed to secure hash algorithm 110. The result of secure hash algorithm 110 is then used as input to universal hash algorithm 305, and from the result derived key 115-1 can be generated.

Whereas sequence 405 shows secure hash algorithm 110 being used before universal hash algorithm 305, sequence 410 reverses this ordering. Thus, master key 105 is used as input to universal hash algorithm 305. The result of universal hash algorithm 305 is then used as input to secure hash algorithm 110, from which result derived key 115-1 can be generated.

Secure hash algorithm 110 and universal hash algorithm 305 can be implemented in any desired form. For example, secure hash algorithm 110 and universal hash algorithm 305 can be implemented in any variety of Read Only Memory (ROM), in firmware, or as software stored in a memory, to provide a few examples where the implementations of secure hash algorithm 110 and universal hash algorithm 305 are executed by general purpose processors. Implementations can also include dedicated devices: for example, a processor can be specifically designed to implement secure hash algorithm 110 and universal hash algorithm 305. Thus, as another example, a calculator can be designed to implement either secure hash algorithm 110 or universal hash algorithm 305. A person skilled in the art will recognize other ways in which secure hash algorithm 110 and universal hash algorithm 305 can be implemented.

FIG. 5 shows a server and device capable of performing data transformations, key generation, key wrapping, and data encryption, according to an embodiment of the invention. In FIG. 5, server 505 is shown. Server 505 includes data transformer 510, key derivation function 515, key wrapping function 520, and encryption function 525. Data transformer 510 is responsible for performing a data transformation. As will be discussed below with reference to FIGs. 8-9, 10A-10C, and 11, data transformations, while intrinsically not secure, increase the complexity of encoded data by scrambling the data, thereby making

cryptanalysis more difficult. For example, data transformation can mask patterns that exist in the encoded, but not transformed, data.

Key derivation function 515 is responsible for deriving keys for use in encrypting data. Although it is true that any key can be used to encrypt data, the more a particular key is used, the more likely it is that the key can be determined with cryptanalysis. Thus, some systems rely on a master key to generate derived keys, which are then used to encrypt the data. As often as desired, a new derived key can be generated; any data encrypted using only derived keys will then provide no value in breaking messages encrypted with the new derived key. Existing key derivation functions exist; three new key derivation functions are described below with reference to FIGs. 12-13 and 15-16.

Key wrapping function 520 is responsible for wrapping a key for transmission. Key wrapping is typically accomplished by encrypting the key for transmission. As an example, RSA can be used to encrypt (that is, wrap) the key. The key, now sufficiently secured, can be transmitted, even over insecure connections, to other machines, where the key can be unwrapped (decrypted) and used for data encryption/decryption.

Often, the wrapped key is a key for use with a private key, or symmetric, cryptosystem, which is wrapped using a public key, or asymmetric, cryptosystem. A private key cryptosystem is one where the same key is used to encrypt and decrypt, as opposed to a public key cryptosystem, which use different keys to encrypt and decrypt. For example, DES and AES are private key cryptosystems; RSA is a public key cryptosystem. While public key cryptosystems make it possible to safely distribute a key (there is no worry that the key can be intercepted and used by a third party to decrypt private messages), public key cryptosystems often are slower to implement and result in longer messages than private key cryptosystems. Obviously, to wrap a key using a public key cryptosystem, server 505 needs to know the public key of the device to which the wrapped key is to be communicated. But a person skilled in the art will recognize that any encryption algorithm can be used to wrap the key, and that the key to be wrapped can be for any kind of cryptosystem.

Encryption function 525 is used to encrypt data. Typically, the data is encrypted using the key that is wrapped using key wrapping function 520, although a person skilled in the art will recognize that any key can be used to encrypt the data, that the data can be any data that is desired to be encrypted, and that any desired encryption function can be used.

FIG. 5 also shows device 530 capable of performing data transformations, key wrapping, and data encryption, according to an embodiment of the invention. Despite the fact that device 530 looks like a personal digital assistant (PDA), a person skilled in the art

will recognize that device 530, as well as server 505, can be any device using security algorithms. Thus, for example, device 530 might be a computer (e.g., a desktop or notebook computer), exchanging files with server 505 (which might be an ordinary computer, and not a server per se). Or, device 530 might be digital media device: e.g., to present digital content to a user, with server 505 providing the content to device 530. Alternatively, device 530 might receive the content from any legitimate source, with server 505 specifying the rights granted to device 530 with respect to the content. Or, device 530 might be software to implement some functionality stored on some medium used with a general-purpose machine, such as a computer. In this variation, what makes device 530 part of the system shown in FIG. 5 is less dependent on the hardware of device 530, and more dependent on the software being executed by device 530. A person skilled in the art will recognize that the software can implement any desired functionality, and that the software can be stored on any appropriate medium, such as a floppy disk, any variety of compact disc (CD) or digital video disc (DVD, sometimes also called a digital versatile disc), a tape medium, or a Universal Serial Bus (USB) key, to name a few of the more popular possibilities. Or, device 530 might be a cellular telephone and server 505 a base station, where the cellular telephone and the base station are communicating in an encrypted manner. A person skilled in the art will recognize other variations for device 530 and server 505, and will also recognize that the manner in which server 505 and device 530 communicate can be any manner of communications channel: e.g., wireline, wireless, or any other form of communication.

Device 530 is similar to server 505 of FIG. 5, in that includes data transformer 510, key wrapping function 520, and encryption function 525. Note that unlike server 505 of FIG. 5, device 530 does not include key derivation function 515. This is because key derivation is generally only needed on server 505. Provided there is a way to communicate with the other device, only one device needs to generate the derivative key. Of course, if there is no way to securely communicate the derivative key but both devices can accurately generate the same derivative key, then device 530 can include key derivation function 515 (although then device 530 might not need key wrapping function 520).

FIG. 6 shows a data security device operable to enhance security by using a data transformer in combination with a key wrapper, key deriver, or an encryption function, according to an embodiment of the invention. Data security device 605 can be part of either server 505 or device 530 of FIG. 5, with modification as needed to add or remove components. In data security device 605, input port 610 is responsible for receiving data. The data can be a master key from which to generate a derivative key, a key to be wrapped,

or data to be encrypted, among other possibilities. Divider 615 is responsible for dividing the data into blocks. As discussed below with reference to FIGs. 12-13 and 14-16, sometimes the functions apply data transformations to multiple portions of the data; divider 615 breaks the data up into blocks of the desired sizes so that data transformer 510 can be applied to each
5 block. Data transformer 510 is responsible for performing the data transformation, which is discussed further below with reference to FIGs. 12-13 and 14-16. Combiner 620 is responsible for combining the blocks, after their data transformation, back together for application of the appropriate security function. Various security functions that can be used include key derivation function 515, key wrapping function 520, or encryption function 525.
10 Finally, output port 625 outputs the data, after transformation and/or application of the security function.

It is worth noting that, although typically divider 615 breaks the data into blocks that conform to the size of the data transformation algorithm, this is not required. Thus, divider 615 might break the data up into blocks that are smaller or larger than the expected input to
15 data transformer 510. If divider 615 breaks the data up into blocks that are smaller than expected by data transformer 510, the data can be padded to make them large enough; if divider 615 breaks the data up into blocks larger than expected by data transformer 510, data transformer 510 can apply the data transformation to only as many bits of the data as it needs. For example, if data transformer 510 is implemented as described in the embodiment of FIG.
20 10, data transformer 510 operates on 8 byte inputs. If data transformer 510 receives more than 8 bytes, data transformer 510 can apply to only 8 bytes of the input. These can be any 8 bytes within the data: e.g., the first 8 bytes, the last 8 bytes, or any other desired combination.

It is also worth noting that any data can be transformed. Thus, the data to be transformed can be a master key, where the transformed master key is to be used to generate
25 derivative keys. Or, the data can be a derivative key that is to be wrapped before transmission. Or, the data can be data that is to be encrypted using an implementation of an encryption algorithm. A person skilled in the art will recognize other types of data that can be transformed.

FIGs. 7A-7B show a flowchart for using the data security device of FIG. 6, according
30 to an embodiment of the invention. In FIG. 7A, at block 705, the data is divided into blocks. At block 710, each of the blocks can be transformed using a data transformation. Each of blocks can be independently data transformed or not, as desired; in other words, some blocks might be transformed, and others not. At block 715, the blocks can be reassembled. As shown by dashed line 720, blocks 705-715 are optional, and can be skipped if not needed.

In FIG. 7B, the data security device can be used in different ways. At block 725, a key wrapping algorithm can be applied to the data. At block 730, a key derivation algorithm can be applied to the data. And at block 735, a data encryption algorithm can be applied to the data.

FIG. 8 shows details of the data transformer of FIGs. 5 and 6, according to an embodiment of the invention. In the embodiment of data transformer 510 shown in FIG. 8, data transformer 510 operates by permuting bit groups using permutation functions. Data transformer 510 includes input port 805 to receive data to be transformed, divider 810, padder 815, permuter 820, and output port 825 to output the transformed data. Divider 810 is responsible for dividing the input data into the bit groups for application of the permutation functions. In fact, divider 810 starts by dividing the data into two segments. The first segment includes bits that are used to control the application of the permutation functions on the bit groups, which are portioned from the second segment. In one embodiment, the data includes 64 bits; the first segment includes 8 bits, and the second segment includes 8 7-bit groups. But a person skilled in the art will recognize that the data can be of any length, and the data can be divided into groups of any desired lengths, even with different groups being of different length. Finally, the first segment, which includes the bits that control the application of the permutation groups, can be omitted, if the individual groups are always permuted.

If data transformer 510 supports receiving data of unpredictable sizes (instead of assuming that the data is always of a fixed size), then divider 810 might not be able to divide the data into bit groups properly. Padder 815 can be used to pad the data with additional bits, so that the data is of appropriate length to be properly divided.

In one embodiment, the application of the permutation functions is controlled by the bits of the first segment: a bit group is permuted using a particular permutation function if a corresponding bit in the first segment is set. For example, if the corresponding bit has the value of 1, then the corresponding group is permuted using the appropriate permutation function; if the corresponding bit has the value 0, then the corresponding group is not permuted. Alternatively, if the corresponding bit has the value 0, the corresponding bit group can be viewed as having been permuted using the identity permutation function. The permutation functions can be indexed as well; if the number of permutation function matches the number of bit groups in the second segment (and therefore also matches the number of bits in the first segment), then a single index can identify three corresponding elements: a bit

in the first segment, a bit group in the second segment, and a permutation function to apply to the bit group.

Permuter 820 is responsible for controlling the permutation of the bit groups of the second segment. In one embodiment, permuter 820 implements permutations according to the functions shown in Table 1 below, although a person skilled in the art will recognize that any permutation functions can be used.

Table 1

Function	Permutation (of a b c d e f g)
P ₁	f a e b d g c
P ₂	g f d a b c e
P ₃	c g b f a e d
P ₄	e c a g f d b
P ₅	d e f c g b a
P ₆	b d g e c a f
P ₇	e c a g f d b
P ₈	c g b f a e d

There are some interesting features of the permutations shown in Table 1. First, each of the permutation functions is a power of permutation function P₁. Thus, P₂ = P₁ ◦ P₁, P₃ = P₂ ◦ P₁ (= P₁ ◦ P₁ ◦ P₁), etc. Because P₆ ◦ P₁ would result in P₁ again, P₇ and P₈ are chosen to repeat earlier powers of P₁. This means that data transformer 510 only needs to know the implementation of one permutation function; the rest of the permutation functions can be derived from the base permutation function. Second, the permutations of Table 1 do not introduce any structures in the data that are similar to those found in encryption functions such as RSA, DES, AES, SHA-1, etc.

Because permutation functions are invertible, the data transformation that results from applying the permutation functions of Table 1 is easily reversible. Table 2 shows the permutation functions that are the inverses of the permutation functions of Table 1.

Table 2

Function	Permutation (of a b c d e f g)
P ₁ ⁻¹	b d g e c a f
P ₂ ⁻¹	d e f c g b a
P ₃ ⁻¹	e c a g f d b
P ₄ ⁻¹	c g b f a e d
P ₅ ⁻¹	g f d a b c e
P ₆ ⁻¹	f a e b d g c
P ₇ ⁻¹	c g b f a e d
P ₈ ⁻¹	e c a g f d b

Thus, to reverse the data transformation applying the permutation functions of Table 1, all that is needed is to apply a second data transformation, using the permutation functions of Table 2. To make this reverse transformation possible, output port 825 outputs the bits of the first segment directly, along with the permuted groups; otherwise, a receiver of the

transformed data would not know which bit groups have been permuted.

As with the permutation functions of Table 1, all of the permutation functions in Table 2 can be derived from a single base function: in this case, P_6^{-1} . Thus, $P_5^{-1} = P_6^{-1} \circ P_6^{-1}$, $P_4^{-1} = P_5^{-1} \circ P_6^{-1} (= P_6^{-1} \circ P_6^{-1} \circ P_6^{-1})$, etc.

FIG. 9 shows details of the data transformer of FIGs. 5 and 6, according to another embodiment of the invention. In FIG. 9, input port 905 and output port 910 operate similarly as in data transformer 510 of FIG. 8. But rather than permuting the data using permutation functions, data transformer 510 of FIG. 9 operates by computing an exponential permutation on the data: this calculation is done by calculator 915. In one embodiment, data transformer 510 operates on data input that is 3 bytes long. The first segment is used to calculate a power, to which the last two bytes are raised. The result is then taken modulo a modulus.

For example, one embodiment computes the data transformation as

$Y = ((B + 1)^{(2A + 1)} \bmod 65537) - 1$, where A is the first byte of the data input and B is the last two bytes of the data input. The transformed data then includes A and Y , and is 3 bytes long.

But a person skilled in the art will recognize that the input can be of different lengths, and

that different exponential permutation functions can be applied.

The above-shown exponential permutation function has some advantages. First, abstract algebra shows that where the exponent and the modulus (minus one) are relatively prime, the function cycles through all possible values between 1 and the modulus, which means that the exponential permutation function is a permutation. By selecting 65537 as the prime number, one less than 65537 is 65536, which is a power of 2. Thus, regardless of the value of A , $(2A + 1)$ is odd, and is therefore relatively prime to 65536. Second, if A is 0, then the data output is unchanged. Finally, as with the permutation data transformer of FIG. 8, the structure of data transformer 510 of FIG. 9 uses a structure not existing in cryptographic algorithms such as RSA, DES, AES, SHA-1, etc.

If data transformer 510 supports receiving data of unpredictable sizes (instead of assuming that the data is always of a fixed size), then divider 920 might not be able to divide the data into segments of appropriate size. Padder 925, as with padder 815 in the data transformer of FIG. 8, can be used to pad the data with additional bits, so that the data is of appropriate length to be properly divided.

As with the permutation data transformer of FIG. 8, data transformer 510 of FIG. 9 is reversible. To make it possible to reverse the data transformation, output port 910 outputs A unchanged along with Y . Then, to reverse the exponential permutation, calculator 915
5 computes the inverse of $2A + 1$ modulo 65536 (that is, 65537-1). If this inverse is called e , then the reverse exponential permutation is $((Y + 1)^e \bmod 65537) - 1$. The result of this calculation restores the original bytes B . Thus, the exponential permutation can be reversed simply by applying a second data transformation, changing the exponent of the data transformer.

Now that the apparatuses of FIGs. 8 and 9 have been presented, the methods of their
10 use can be understood. FIGs. 10A-10C show a flowchart for using the data transformer of FIG. 8, according to an embodiment of the invention. In FIG. 10A, at block 1005, the data is received. At block 1010, the data is divided into two segments (assuming that the permutation of bit groups are controlled by bits in the first segment). At block 1015, the data transformer checks to see if the second data segment can be divided evenly into groups. If
15 not, then at block 1020 the data is padded to support dividing the second segment into evenly-sized groups. (This assumes that the data transformer attempts to divide the data input into evenly-sized groups; if the data transformer does not need to divide the input data into evenly-sized groups, then blocks 1015 and 1020 can be omitted.)

At block 1025 (FIG. 10B), the second segment is divided into bit groups. Although
20 block 1025 describes the second segment as being divided into groups of equal size, as described above, the groups can be divided into groups of unequal size, if the data transformer supports this. At block 1030, each group is associated with a bit in the first segment. At block 1035, a base permutation function is defined. At block 1040, other permutation functions are defined as powers of the base permutation function. (Again, there
25 is no requirement that the permutations be powers of a base permutation function; each of the permutation functions can be unrelated to the others, in which case blocks 1035 and 1040 can be modified/omitted.) At block 1045, the permutation functions are indexed.

At block 1050 (FIG. 10C), the data transformer checks to see if any bits in the first segment (which controls the application of the permutation functions to the bit groups in the
30 second segment) have yet to be examined. If there are unexamined bits, then at block 1055 the data transformer examines the bit to see if it is set. If the bit is set, then at block 1060 the permutation function indexed by the bit is identified, and at block 1065 the identified permutation is applied to the associated permutation group. Control then returns to block 1050 to see if there are any further unexamined bits in the first segment. After all bits in the

first segment have been examined, then at block 1070 the data transformer constructs the data transformation from the first segment and the permuted bit groups.

FIG. 11 shows a flowchart for using the data transformer of FIG. 9, according to an embodiment of the invention. At block 1105, the data transformer receives the data. At block 1110, the data transformer divides the data into two segments. At block 1115, the first segment is used to construct a power that is relatively prime to the selected modulus. At block 1120, the second segment is raised to the computed power. At block 1125, the remainder is computed by taking the result modulo the modulus. Finally, at block 1130, the data transform is constructed from the first segment and the remainder.

As discussed above with reference to FIG. 5, existing key derivation functions exist. But the existing key derivation functions do not provide the advantages of both the secure hash function and the universal hash function, as described above with reference to FIG. 4. FIG. 12 shows details of one key derivation function that combine the advantages of a secure hash function and a universal hash function. In FIG. 12, key derivation function 515 includes input port 1205 and output port 1210, which are used to provide the inputs to the key derivation function and the output derived key, respectively. Key derivation function 515 also includes divider 1215, combiner 1220, hash 1225, determiner 1230, calculator 1235, and bit selector 1240.

Divider 1215 divides the master key into two parts. Combiner combines the first part of the master key with a counter, which can be part of the input data. One way to combine the master key with the counter is by concatenating the first part of the master key with the counter, which can be of any size (e.g., 4 bytes). This concatenation can be performed in either order: that is, either the first part of the master key or the counter can be the front of the combination. The result of this combination is then hashed using hash function 1225, which can be a secure hash function. (In this embodiment, hash function 1225 takes the place of secure hash algorithm 110 in sequence 405 of FIG. 4.)

Determiner 1230 is used to determine two numbers from the second part of the master key. In one embodiment, these two numbers, a and b , are determined as the first and last 32 bytes of the second part of the master key, modulo a prime number p . Selecting a and b in this manner calls for the master key to be of sufficient length for the second part of the master key to be 64 bytes long. But a person skilled in the art will recognize that the master key does not necessarily have to be this long. For example, if computing a and b modulo p sufficiently alters the bits of a and b , a and b might be selected in such a way that their original bits overlap from within the second part of the master key.

A particular choice for the prime number can be $p_{192} = 2^{192} - 2^{64} - 1$, although a person skilled in the art will recognize that other primes can be selected instead. Calculator 1235 can then implement the universal hash function of $ax + b \bmod p$, where x is the result of hash 1225. (This universal hash function takes the place of universal hash algorithm 305 in sequence 405 of FIG. 4.) Finally, bit selector 1240 selects the bits from the result of the universal hash function for the derived key, which can then be output. For example, bit selector 1240 can select the least significant bits of the result of the universal hash function as the derived key.

FIG. 13 shows details of the key derivation function of FIGs. 5 and 6, according to another embodiment of the invention. In contrast to the embodiment of the invention shown in FIG. 12, which implements a key derivation function according to sequence 405 of FIG. 4, key derivation function 515 of FIG. 13 does not apply the universal hash algorithm after the secure hash algorithm. Instead, the embodiment of the invention shown in FIG. 13 applies a liner mapping to the input to the secure hash algorithm.

As with key derivation function 515 of FIG. 12, key derivation function 515 of FIG. 13 includes input port 1305 and output port 1310, which receive the master key as input and output the derived key, respectively. Key derivation function 515 of FIG. 13 also includes divider 1315, encoder 1320, combiner 1325, hash 1330, and bit selector 1335.

Divider 1315, as with divider 1215 of FIG. 12, divides the master key into two parts. Encoder 1320 then encodes a counter. Encoder 1320 can operate in any manner desired. For example, encoder 1320 can operate by repeating the counter to extend it to the length of the first part of the master key. So, for example, if the first part of the master key is 64 bytes long and the counter is represented using 4 bytes, encoder 1320 can repeat those 4 bytes 16 times, to extend the counter to a 64 byte length. Combiner 1325 can then combine the encoded counter with each part of the master key separately. For example, combiner 1325 can combine the parts of the master key and the encoded counter at the bit level. One embodiment uses an XOR binary function to combine the parts of the master key and the encoded counter. But a person skilled in the art will recognize that combiner 1325 can use any bitwise binary function, or indeed any function, to combine the parts of the master key and the encoded counter. Combiner 1325 can then recombine the two parts of the master key (after the combination with the encoded counter) back together: for example, the two parts can be concatenated together (but a person skilled in the art will recognize that combiner 1325 can recombine the two parts of the master key in other ways). Combiner 1325 can also concatenate the recombined parts of the master key with the encoded counter one more time.

Hash 1330 takes the output of combiner 1325 and hashes it. Hash 1330 can be a secure hash function. Bit selector 1335, as with bit selector 1240 in FIG. 12, can then select bits from the result of hash 1330 as the derived key.

Now that the apparatuses of FIGs. 12 and 13 have been presented, the methods of their use can be understood. FIG. 14 shows a flowchart for using the key derivation function of FIG. 12, according to an embodiment of the invention. At block 1405, the master key is divided into segments. At block 1410, the first segment is combined with an encoded counter. As described above with reference to FIG. 12, this combination can be the concatenation of the first segment with the encoded counter. At block 1415, the combined first segment is hashed.

At block 1420, two numbers are determined from the second segment. As discussed above with reference to FIG. 12, these two numbers can be determined relative to a modulus. At block 1425, a universal hash function is defined using the two determined numbers and the modulus. At block 1430, the result of the hash is applied to the universal hash function. At block 1435, bits are selected from the result of the universal hash as the derivative key.

FIG. 15 shows a flowchart for using the key derivation function of FIG. 13, according to an embodiment of the invention. At block 1505, the master key is divided into segments. At block 1510, each of the segments is combined with an encoded counter. As described above with reference to FIG. 13, this can be done by applying an XOR bit function to each of the segments individually with the encoded counter. At block 1515, the combined blocks are then recombined, and (as discussed above with reference to FIG. 13), can also be combined again with the encoded counter. At block 1520, this modified master key is then hashed, and at block 1525, bits are selected from the result of the hash as the derivative key.

The key derivation functions shown in FIGs. 12-15 are only two examples. Other key derivation functions can also be used that combine the advantages of a secure hash algorithm and a universal hash algorithm. FIG. 16 shows a flowchart for yet another key derivation function in the data security device of FIG. 5, according to an embodiment of the invention. At block 1605, the master key is divided into segments. At block 1610, the segments are transformed using data transformation. Because the segments will typically be larger than the data transformer can use, only a subset of the segments are used: e.g., only the first bytes needed by the data transformation. At block 1615, the transformed segments are combined, and combined with encoded counter: e.g., the segments and the encoded counter can be concatenated together. At block 1620, the result is hashed, and at block 1625, bits are selected from the result of the hash as the derivative key.

While the apparatuses of FIGs. 12-13, and the flowcharts of FIGs. 14-16 show the generation of a single derivative key from a master key, it is worth noting that embodiments of the invention can easily be adapted to generate repeated derivative keys. These additional derivative keys can be generated in numerous ways. For example, the flowcharts of FIGs.

5 1416 all include counters. For each additional derivative key desired, the counter can be incremented. Thus, to derive the first key, the counter can use the value 1, to derive the second key, the counter can use the value 2, and so on.

In another variation, rather than using bit selector 1240 of FIG. 12 or bit selector 1335 of FIG. 13 to select bits for the derivative key, enough results can be generated at one time to
10 select bits from the combined results for all the derivative keys. For example, assume that u keys are desired, each k bits long, and further assume that the results of the apparatuses of FIGs. 12-13 and/or the flowcharts of FIGs. 14-16 produce l bits before bit selection. If the key derivation function is applied m times, so that $m * l \geq u * k$, then the u derivative keys can all be selected at the same time from the $m * l$ resulting bits. For example, the $m * l$
15 resulting bits might all be concatenated together; the first key might then be selected as the first k bits, the second key might be selected as the second k bits, and so on until all u keys have been selected.

The following discussion is intended to provide a brief, general description of a suitable machine in which certain aspects of the invention may be implemented. Typically,
20 the machine includes a system bus to which is attached processors, memory, e.g., random access memory (RAM), read-only memory (ROM), or other state preserving medium, storage devices, a video interface, and input/output interface ports. The machine may be controlled, at least in part, by input from conventional input devices, such as keyboards, mice, etc., as well as by directives received from another machine, interaction with a virtual reality (VR)
25 environment, biometric feedback, or other input signal. As used herein, the term "machine" is intended to broadly encompass a single machine, or a system of communicatively coupled machines or devices operating together. Exemplary machines include computing devices such as personal computers, workstations, servers, portable computers, handheld devices, telephones, tablets, etc., as well as transportation devices, such as private or public
30 transportation, e.g., automobiles, trains, cabs, etc.

The machine may include embedded controllers, such as programmable or non-programmable logic devices or arrays, Application Specific Integrated Circuits, embedded computers, smart cards, and the like. The machine may utilize one or more connections to one or more remote machines, such as through a network interface, modem, or other

communicative coupling. Machines may be interconnected by way of a physical and/or logical network, such as an intranet, the Internet, local area networks, wide area networks, etc. One skilled in the art will appreciate that network communication may utilize various wired and/or wireless short range or long range carriers and protocols, including radio
5 frequency (RF), satellite, microwave, Institute of Electrical and Electronics Engineers (IEEE) 802.11, Bluetooth, optical, infrared, cable, laser, etc.

The invention may be described by reference to or in conjunction with associated data including functions, procedures, data structures, application programs, etc. which when accessed by a machine results in the machine performing tasks or defining abstract data types
10 or low-level hardware contexts. Associated data may be stored in, for example, the volatile and/or non-volatile memory, e.g., RAM, ROM, etc., or in other storage devices and their associated storage media, including hard-drives, floppy-disks, optical storage, tapes, flash memory, memory sticks, digital video disks, biological storage, etc. Associated data may be delivered over transmission environments, including the physical and/or logical network, in
15 the form of packets, serial data, parallel data, propagated signals, etc., and may be used in a compressed or encrypted format. Associated data may be used in a distributed environment, and stored locally and/or remotely for machine access.

Having described and illustrated the principles of the invention with reference to illustrated embodiments, it will be recognized that the illustrated embodiments may be
20 modified in arrangement and detail without departing from such principles. And, though the foregoing discussion has focused on particular embodiments, other configurations are contemplated. In particular, even though expressions such as “in one embodiment” or the like are used herein, these phrases are meant to generally reference embodiment possibilities, and are not intended to limit the invention to particular embodiment configurations. As used
25 herein, these terms may reference the same or different embodiments that are combinable into other embodiments.

Consequently, in view of the wide variety of permutations to the embodiments described herein, this detailed description and accompanying material is intended to be illustrative only, and should not be taken as limiting the scope of the invention. What is
30 claimed as the invention, therefore, is all such modifications as may come within the scope and spirit of the following claims and equivalents thereto.

CLAIMS

1. A data transformer, comprising:
an input port to receive data;
a divider to divide said data into a first segment and a second segment and to divide
5 said second segment into at least one group;
a permuter including an implementation of a permutation function to permute at least
one of said groups into a permuted group according to a corresponding bit in said first
segment; and
an output port to output said first segment and at least said permuted group as
10 transformed data.
2. A data transformer according to claim 1, wherein the divider is operative to
divide said second segment into groups so that a number of said groups in said second
segment equals a number of bits in said first segment.
15
3. A data transformer according to claim 1, wherein the divider is operative to
divide said second segment into said groups, each of said groups having a predefined size.
4. A data transformer according to claim 3, wherein the divider includes a padder
20 to pad said second segment so that each of said groups have said predefined size.
5. A data transformer according to claim 1, wherein the permuter includes
implementations of at least two permutation functions.
- 25 6. A data transformer according to claim 5, wherein a number of permutation
functions is equal to a number of said groups in said second segment.
7. A data transformer according to claim 6, wherein the permuter is operative to
permute each of said groups using one of said permutation functions according to the
30 corresponding bit in said first segment.
8. A data transformer according to claim 5, wherein the implementations of said
permutation functions includes:
an implementation of a base permutation function; and

implementations of powers of the base permutation function.

9. A data security device, comprising:

a data transformer, including:

5 an input port to receive data;

a divider to divide said data into a first segment and a second segment and to divide said second segment into at least two groups, each group having a predefined size, so that a number of groups in said second segment equals a number of bits in said first segment;

10 a permuter including an implementation of a permutation function to permute at least one of said groups into a permuted group according to a corresponding bit in said first segment; and

an output port to output said first segment and at least said permuted group as transformed data; and

15 an implementation of a security algorithm to secure said transformed data.

10. A data security device according to claim 9, wherein:

said data includes a master key; and

20 the implementation of a security algorithm includes an implementation of a key derivation function to use said transformed data to generate a derivative key of said master key.

11. A data security device according to claim 9, wherein:

said data includes a key to be wrapped; and

25 the implementation of said security algorithm includes an implementation of a key wrapping function to wrap said transformed data.

12. A data security device according to claim 11, wherein the implementation of

30 said key wrapping function includes an implementation of RSA to wrap said transformed data.

13. A data security device according to claim 9, wherein the implementation of

said security algorithm includes an implementation of an encryption algorithm to use said transformed data to encrypt said data.

14. A data security device according to claim 13, wherein the implementation of said security algorithm includes an implementation of AES to use said transformed data to encrypt said data.

5

15. A data security device according to claim 9, further comprising a second divider to divide an input into at least two blocks, the data transformer operative separately on each block.

10

16. A data security device according to claim 15, further comprising a combiner to combine a result of the data transformer on each block into a single transformed data to be secured by the implementation of said security algorithm.

15

17. A data security device according to claim 9, wherein the permuter includes implementations of at least two permutation functions, a number of permutation functions is equal to a number of said groups in said second segment.

20

18. A data security device according to claim 17, wherein the permuter is operative to permute each of said groups using one of said permutation functions according to the corresponding bit in said first segment.

19. A data security device according to claim 17, wherein the implementations of said permutation functions includes:

25

an implementation of a base permutation function; and
implementations of powers of the base permutation function.

30

20. A method for generating a data transform, comprising:
receiving data;
dividing the data into a first segment and a second segment, each of the first segment and the second segment including at least one bit;
organizing the bits in the second segment into at least one group;
associating each of the groups with a bit in the first segment;
applying a permutation function to at least one of the groups according to the associated bit in the first segment; and

constructing the data transform from the first segment and at least the permuted groups.

21. A method according to claim 20, wherein organizing the bits in the second
5 segment includes organizing the bits in the second segment into a number of groups, the number of groups equal to a number of bits in the first segment.

22. A method according to claim 20, wherein organizing the bits includes
10 organizing the bits in the second segment into at least two groups, each group having a same number of bits.

23. A method according to claim 22, wherein organizing the bits includes padding
the data so that the second segment can be organized into the groups, each of the groups
including the same number of bits.

24. A method according to claim 20, further comprising defining the permutation
function.

25. A method according to claim 24, wherein defining the permutation function
20 includes defining a number of permutation functions, the number of permutation functions equal to a number of the groups.

26. A method according to claim 25, wherein applying a permutation function
includes applying a different permutation function to each of the groups according to the
25 corresponding bit in the first segment.

27. A method according to claim 26, wherein:
defining a number of permutation functions includes assigning an index to each of the
permutation functions; and
30 applying a different permutation function includes selecting the permutation function with the index corresponding to a number for the bit in the first segment associated with the group.

28. A method according to claim 25, wherein defining a number of permutation functions includes:

defining a base permutation function; and

defining each of the remaining permutation functions as a power of the base

5 permutation function.

29. A method for enhancing security of data, comprising:

transforming the data, including:

receiving the data;

10 dividing the data into a first segment and a second segment, each of the first segment and the second segment including at least one bit;

organizing the bits in the second segment into a number of groups, the number of groups equal to a number of bits in the first segment;

associating each of the groups with a bit in the first segment;

15 applying a permutation function to at least one of the groups according to the associated bit in the first segment; and

constructing the data transform from the first segment and at least the permuted groups; and

20 applying an implementation of a security algorithm to the data transform to secure the data transform.

30. A method according to claim 29, wherein:

receiving the data includes receiving a master key from which to generate a derivate key; and

25 applying an implementation of a security algorithm includes applying an implementation of a key derivation function to the data transform to generate the derivative key of the master key.

31. A method according to claim 30, wherein applying an implementation of a key derivation function includes:

30 combining the transformed data with an encoded counter to produce a combined result;

securely hashing the combined result to produce a hash; and

selecting a subset of bits in the hash as the derivative key.

32. A method according to claim 30, wherein:

transforming the data further includes combining the master key with a counter to produce the data; and

5 applying an implementation of a key derivation function includes:
 securely hashing the transformed data to produce a hash; and
 selecting a subset of bits in the hash as the derivative key.

33. A method according to claim 29, wherein:

10 receiving the data includes receiving a key to be wrapped as the data; and
 applying an implementation of a security algorithm includes applying an
implementation of a key wrapping function to the data transform to wrap the key.

34. A method according to claim 33, wherein applying an implementation of a key

15 wrapping function includes applying an implementation of RSA to the data transform to wrap
the key.

35. A method according to claim 29, wherein applying an implementation of a

20 security algorithm includes applying an implementation of an encryption algorithm using the
data transform as a key to encrypt the data.

36. A method according to claim 35, wherein applying an implementation of an

25 encryption algorithm includes applying an implementation of AES using the data transform
as the key to encrypt the data.

37. A method according to claim 29, wherein organizing the bits includes

organizing the bits in the second segment into the number of groups, the number of groups
equal to the number of bits in the first segment, each group having a same number of bits.

38. A method according to claim 29, further comprising defining the permutation

30 function.

39. A method according to claim 38, wherein defining the permutation function includes defining a number of permutation functions, the number of permutation functions equal to the number of groups.

5 40. A method according to claim 39, wherein defining a number of permutation functions includes:

defining a base permutation function; and
defining each of the remaining permutation functions as a power of the base permutation function.

10 41. A method according to claim 29, further comprising:
dividing an input into at least two blocks, transforming each block separately; and
combining a result of the data transformation on each block into a single transformed data to be secured by the application of the implementation of the security algorithm.

15 42. A data transformer, comprising:
an input port to receive data including a plurality of bits;
a divider to divide said data into a first segment and a second segment;
a calculator to compute an exponential permutation of said data using said first
20 segment, said second segment, and a predefined modulus; and
an output port to output said first segment and said exponential permutation as transformed data.

25 43. A data transformer according to claim 42, wherein the calculator includes an implementation of a first formula to compute a power as a function of said first segment, said power being relatively prime to a function of said predefined modulus.

30 44. A data transformer according to claim 43, wherein the calculator further includes an implementation of a second formula to compute a result of raising a function of said second segment to said power.

45. A data transformer according to claim 44, wherein the calculator further includes an implementation of a third formula to compute said exponential permutation as said result modulo said predefined modulus.

46. A data security device, comprising:

a data transformer, including:

an input port to receive data including a plurality of bits;

a divider to divide said data into a first segment and a second segment;

a calculator to compute an exponential permutation of said data using said first segment, said second segment, and a predefined modulus, including:

an implementation of a first formula to compute a power as a function of said first segment, said power being relatively prime to a function of said predefined modulus;

an implementation of a second formula to compute a result of raising a function of said second segment to said power; and

an implementation of a third formula to compute said exponential permutation as said result modulo said predefined modulus; and

an output port to output said first segment and said exponential permutation as transformed data; and

an implementation of a security algorithm to secure said transformed data.

47. A data security device according to claim 46, wherein:

said data includes a master key; and

the implementation of a security algorithm includes an implementation of a key derivation function to use said transformed data to generate a derivative key of said master key.

48. A data security device according to claim 46, wherein:

said data includes a key to be wrapped; and

the implementation of said security algorithm includes an implementation of a key wrapping function to wrap said transformed data.

49. A data security device according to claim 48, wherein the implementation of

said key wrapping function includes an implementation of RSA to wrap said transformed data.

50. A data security device according to claim 46, wherein the implementation of said security algorithm includes an implementation of an encryption algorithm to use said transformed data as a key to encrypt said data.

5 51. A data security device according to claim 50, wherein the implementation of said security algorithm includes an implementation of AES to use said key to encrypt said data.

10 52. A data security device according to claim 46, further comprising a second divider to divide an input into at least two blocks, the data transformer operative separately on each block.

15 53. A data security device according to claim 52, further comprising a combiner to combine a result of the data transformer on each block into a single transformed data to be secured by the implementation of said security algorithm.

54. A method for generating a data transform, comprising:
receiving data, the data including a plurality of bits;
dividing the data into a first segment and a second segment, each of the first segment
20 and the second segment including at least one bit;
computing an exponential permutation using the first segment, the second segment, and a predefined modulus; and
constructing the data transform from the first segment and the exponential permutation.

25

55. A method according to claim 54, wherein computing an exponential permutation includes computing a power as a function of the first segment, the power being relatively prime to a function of the predefined modulus.

30 56. A method according to claim 55, wherein computing an exponential permutation further includes computing a result of raising a function of the second segment to the power.

57. A method according to claim 56, wherein computing an exponential permutation further includes computing the exponential permutation as the result modulo the predefined modulus.

5 58. A method for enhancing security of data, comprising:

transforming the data, including:

receiving data, the data including a plurality of bits;

dividing the data into a first segment and a second segment, each of the first segment and the second segment including at least one bit;

10 computing a power as a function of the first segment, the power being relatively prime to a function of a predefined modulus;

computing a result of raising a function of the second segment to the power;

computing an exponential permutation as the result modulo the predefined modulus; and

15 constructing the data transform from the first segment and the computed exponential permutation; and

applying an implementation of a security algorithm to the data transform to secure the data transform.

20 59. A method according to claim 58, wherein:

receiving the data includes receiving a master key from which to generate a derivative key; and

applying an implementation of a security algorithm includes applying an implementation of a key derivation function to the data transform to generate the derivative

25 key of the master key.

60. A method according to claim 59, wherein applying an implementation of a key derivation function includes:

30 combining the transformed data with an encoded counter to produce a combined result;

securely hashing the combined result to produce a hash; and

selecting a subset of bits in the hash as the derivative key.

61. A method according to claim 58, wherein:

receiving the data includes receiving a key to be wrapped as the data; and
applying an implementation of a security algorithm includes applying an
implementation of a key wrapping function to wrap the data transform.

5 62. A method according to claim 61, wherein applying an implementation of a key
wrapping function includes applying an implementation of RSA to wrap the data transform.

 63. A method according to claim 58, wherein applying an implementation of a
security algorithm includes applying an implementation of an encryption algorithm using the
10 data transform as a key to encrypt the data.

 64. A method according to claim 63, wherein applying an implementation of an
encryption algorithm includes applying an implementation of RSA using the data transform
as the key to encrypt the data.

15 65. A method according to claim 63, further comprising:
dividing an input into at least two blocks, transforming each block separately; and
combining a result of the data transformation on each block into a single transformed
data to be secured by the application of the implementation of the security algorithm.

20 66. An apparatus comprising:
an input port to receive a master key;
an implementation of a universal hash algorithm;
an implementation of a hash algorithm;
25 means for generating a derivative key from said master key using the implementation
of said universal hash algorithm and said hash algorithm; and
an output port to output said derivative key.

 67. An apparatus according to claim 66, wherein the implementation of said
30 universal hash algorithm includes:
a divider to divide said master key into a first segment and a second segment;
a repeater to repeat said counter to form an encoded counter as a longer bit pattern;
an implementation of a first bitwise binary function operative on said first segment
and said encoded counter to produce a first result;

an implementation of a second bitwise binary function operative on said second segment and said encoded counter to produce a second result; and

a combiner to combine said first result, said second result, and said encoded counter to produce said result.

5

68. An apparatus according to claim 66, wherein:

the apparatus further comprises a divider to divide said master key into a first segment and a second segment;

the implementation of said hash algorithm includes:

10

a combiner to combine said first segment with said counter to produce a modified first segment; and

means for using the implementation of said hash algorithm to hash said modified first segment into a hash value; and

the implementation of said universal hash algorithm includes:

15

a determiner to determine a first number and a second number from said second segment;

a calculator including an implementation of an arithmetic formula to compute a result using said hash value, said first number, and said second number; and

a bit selector to select a set of bits from said result as a derivative key.

20

69. An apparatus according to claim 66, wherein the means for generating includes:

means for using the implementation of said universal hash algorithm with said master key to produce a result; and

25

means for using the implementation of said hash algorithm with said result and a counter to produce said derivative key.

70. An apparatus according to claim 69, wherein the implementation of said universal hash algorithm includes:

30

a divider to divide said master key into a first segment and a second segment;

a repeater to repeat said counter to form an encoded counter as a longer bit pattern;

an implementation of a first bitwise binary function operative on said first segment and said encoded counter to produce a first result;

an implementation of a second bitwise binary function operative on said second segment and said encoded counter to produce a second result; and

a combiner to combine said first result, said second result, and said encoded counter to produce said result.

5

71. An apparatus according to claim 66, wherein the means for generating includes:

means for using the implementation of said hash algorithm with said master key and a counter to produce a result; and

10 means for using the implementation of said universal hash algorithm with said result to produce said derivative key.

72. An apparatus according to claim 71, wherein:

15 the apparatus further comprises a divider to divide said master key into a first segment and a second segment;

the means for using the implementation of said hash algorithm includes:

a combiner to combine said first segment with said counter to produce a modified first segment; and

20 means for using the implementation of said hash algorithm to hash said modified first segment into a hash value; and

the means for using the implementation of said universal hash algorithm includes:

a determiner to determine a first number and a second number from said second segment;

25 a calculator including an implementation of an arithmetic formula to compute a result using said hash value, said first number, and said second number; and

a bit selector to select a set of bits from said result as a derivative key.

73. An apparatus comprising:

an input port to receive a master key;

30 a first calculator to implement a universal hash algorithm;

a second calculator to implement a hash algorithm;

a key deriver to generate a derivative key from said master key using the first calculator and the second calculator; and

an output port to output said derivative key.

74. An apparatus according to claim 73, wherein the key deriver includes:
a divider to divide said master key into a first segment and a second segment;
a repeater to repeat said counter to form an encoded counter as a longer bit pattern;
5 a third calculator to implement a first bitwise binary function operative on said first
segment and said encoded counter to produce a first result;
a fourth calculator to implement a second bitwise binary function operative on said
second segment and said encoded counter to produce a second result; and
a combiner to combine said first result, said second result, and said encoded counter
10 to produce said result.

75. An apparatus according to claim 73, wherein:
the apparatus further comprises a divider to divide said master key into a first segment
and a second segment;
15 the second calculator includes:
a combiner to combine said first segment with said counter to produce a
modified first segment; and
a fifth calculator to implement said hash algorithm to hash said modified first
segment into a hash value; and
20 the first calculator includes:
a determiner to determine a first number and a second number from said
second segment;
a sixth calculator to implement an arithmetic formula to compute a result using
said hash value, said first number, and said second number; and
25 a bit selector to select a set of bits from said result as a derivative key.

76. An apparatus according to claim 73, wherein the key deriver includes:
a third calculator to implement said universal hash algorithm with said master key to
produce a result; and
30 a fourth calculator to implement said hash algorithm with said result and a counter to
produce said derivative key.

77. An apparatus according to claim 76, wherein the first calculator includes:
a divider to divide said master key into a first segment and a second segment;

a repeater to repeat said counter to form an encoded counter as a longer bit pattern;
a fifth calculator to implement a first bitwise binary function operative on said first
segment and said encoded counter to produce a first result;
a sixth calculator to implement a second bitwise binary function operative on said
5 second segment and said encoded counter to produce a second result; and
a combiner to combine said first result, said second result, and said encoded counter
to produce said result.

78. An apparatus according to claim 73, wherein the key deriver includes:
10 a third calculator to implement said hash algorithm with said master key and a counter
to produce a result; and
a fourth calculator to implement said universal hash algorithm with said result to
produce said derivative key.

79. An apparatus according to claim 78, wherein:
15 the apparatus further comprises a divider to divide said master key into a first segment
and a second segment;
the third calculator includes:
a combiner to combine said first segment with said counter to produce a
20 modified first segment; and
a third calculator to implement said hash algorithm to hash said modified first
segment into a hash value; and
the fourth calculator includes:
a determiner to determine a first number and a second number from said
25 second segment;
a fifth calculator including an implementation of an arithmetic formula to
compute a result using said hash value, said first number, and said second number;
and
a bit selector to select a set of bits from said result as a derivative key.

80. An apparatus, comprising:
an input port to receive a master key;
a divider to divide said master key into a first segment and a second segment;

a concatenator to concatenate said first segment and a counter to produce a modified first segment;

a hasher to hash said modified first segment into a hash value;

a determiner to determine a first number and a second number from said second

5 segment;

a calculator including an implementation of an arithmetic formula to compute a result using said hash value, said first number, and said second number; and

a bit selector to select a set of bits from said result as a derivative key.

10 81. An apparatus according to claim 80, further comprising an output port to output said derivative key.

82. An apparatus according to claim 80, wherein the calculator includes:

an implementation of a first function to compute a product of said hash value and said

15 first number;

an implementation of a second function to compute a sum of said product and said second number; and

an implementation of a third function to compute said result of said sum modulo a modulus.

20

83. An apparatus according to claim 82, wherein the determiner is operative to determine said first number and said second number modulo said modulus.

84. An apparatus according to claim 82, wherein the implementation of said third

25 function includes an implementation of said third function to compute said result of said sum modulo a prime modulus.

85. An apparatus according to claim 80, wherein the bit selector is operative to

select as said derivative key a set of least significant bits from said result.

30

86. A data security device, comprising:

a key deriver, including:

an input port to receive a master key;

a divider to divide said master key into a first segment and a second segment;

a concatenator to concatenate said first segment and a counter to produce a modified first segment;

a hasher to hash said modified first segment into a hash value;

a determiner to determine a first number and a second number from said second segment modulo a modulus;

a calculator including an implementation of an arithmetic formula to compute a result using said hash value, said first number, and said second number; and

a bit selector to select a set of bits from said result as a derivative key; and an encrypter to encrypt data using said derivative key.

87. A data security device according to claim 86, further comprising a data transformer.

88. A data security device according to claim 87, wherein the data transformer is operative to transform an original master key into said master key.

89. A data security device according to claim 87, wherein the data transformer is operative to transform said derivative key into a transformed derivative key.

90. A data security device according to claim 86, wherein the calculator includes: an implementation of a first function to compute a product of said hash value and said first number;

an implementation of a second function to compute a sum of said product and said second number; and

an implementation of a third function to compute said result from said sum modulo said modulus.

91. A data security device according to claim 86, wherein the implementation of said third function includes an implementation of said third function to compute said result from said sum modulo a prime modulus.

92. A method for performing key derivation, comprising:
hashing a master key to produce a hash value;
determining a first number and a second number from the master key;

computing a universal hash function of the hash value, the first number, and the second number to produce a result; and
selecting a derivative key from bits in the result.

5 93. A method according to claim 92, wherein:
the method further comprises dividing the master key into a first segment and a second segment;
hashing a master key includes hashing the first segment to produce the hash value;
and
10 determining a first number and a second number includes determining the first number and the second number from the second segment.

 94. A method according to claim 93, wherein:
the method further comprises determining a counter; and
15 hashing the first segment includes combining the first segment and the counter.

 95. A method according to claim 92, wherein determining a first number and a second number includes:
deriving a third number and a fourth number from the master key;
20 computing the first number as the third number modulo a modulus; and
computing the second number as the fourth number modulo the modulus.

 96. A method according to claim 92, wherein computing a universal hash function includes:
25 computing a product of the first number and the hash value;
computing a sum of the product and the second number; and
computing the result as the sum modulo a modulus.

 97. A method according to claim 96, wherein computing the result includes
30 computing the result as the sum modulo a prime divisor.

 98. A method according to claim 96, wherein selecting a derivative key includes selecting the derivative key from a set of least significant bits in the result.

99. A method for encrypting a derivative key, comprising:

generating the derivative key, including:

dividing the master key into a first segment and a second segment;

hashing the first segment to produce a hash value;

determining a first number and a second number from the second segment;

computing a product of the first number and the hash value;

computing a sum of the product and the second number; and

computing a result as the sum modulo a modulus; and

selecting the derivative key from bits in the result; and

encrypting data using the derivative key.

100. A method according to claim 99, further comprising applying a data transformation to the master key before generating the derivative key.

101. A method according to claim 100, wherein applying a data transformation includes:

dividing the master key into a third segment and a fourth segment, each of the third segment and the fourth segment including at least one bit;

organizing the bits in the fourth segment into a number of groups, the number of groups equal to a number of bits in the third segment; each group having a same number of bits;

associating each of the groups with a bit in the third segment;

applying a permutation function to at least one of the groups according to the associated bit in the third segment; and

constructing the transformed master key from the third segment and the permuted groups.

102. A method according to claim 100, wherein applying a data transformation includes:

dividing the master key into a third segment and a fourth segment, each of the third segment and the fourth segment including at least one bit;

computing a power as a function of the third segment, the power being relatively prime to a function of a second modulus;

computing a result of raising a function of the fourth segment to the power;

computing an exponential permutation as the result modulo the second modulus; and
constructing the transformed master key from the third segment and the computed
exponential permutation.

5 103. A method according to claim 99, further comprising applying a data
transformation to the derivative key.

104. A method according to claim 99, wherein applying a data transformation
includes:

10 dividing the master key into a third segment and a fourth segment, each of the third
segment and the fourth segment including at least one bit;

 organizing the bits in the fourth segment into a number of groups, the number of
groups equal to a number of bits in the third segment; each group having a same number of
bits;

15 associating each of the groups with a bit in the third segment;

 applying a permutation function to at least one of the groups according to the
associated bit in the third segment; and

 constructing the transformed master key from the third segment and the permuted
groups.

20

105. A method according to claim 103, wherein applying a data transformation
includes:

 dividing the master key into a third segment and a fourth segment, each of the third
segment and the fourth segment including at least one bit;

25 computing a power as a function of the third segment, the power being relatively
prime to a function of a second modulus;

 computing a result of raising a function of the fourth segment to the power;

 computing an exponential permutation as the result modulo the second modulus; and

 constructing the transformed master key from the third segment and the computed

30 exponential permutation.

106. A method according to claim 99, further comprising encrypting the derivative
key.

107. A method according to claim 106, further comprising transmitting the encrypted derivative key.

5 108. A method according to claim 99, further comprising transmitting the encrypted data.

109. A method according to claim 99, wherein:
the method further comprises determining a counter; and
hashing the first segment includes combining the first segment and the counter.
10

110. A method according to claim 99, wherein determining a first number and a second number includes:
deriving a third number and a fourth number from the second segment;
computing the first number as the third number modulo a modulus; and
15 computing the second number as the fourth number modulo the modulus.

111. A method according to claim 99, wherein selecting a derivative key includes selecting the derivative key from a set of least significant bits in the result.

20 112. An apparatus, comprising:
an input port to receive a master key;
a combiner to combine said master key and a value to produce a modified master key;
a hasher to hash said modified master key into a hash value; and
a bit selector to select a set of bits from said hash value as a derivative key.

25 113. An apparatus according to claim 112, wherein:
the apparatus further comprises a repeater to repeat said value to form an encoded value as a longer bit pattern; and
the combiner is operative to combine said master key and said encoded value to
30 produce a modified master key;

114. An apparatus according to claim 112, wherein:
the combiner includes a divider to divide said master key into a first segment and a second segment; and

the combiner is operative to combine said encoded value, said first segment, and said second segment to produce said modified master key.

115. An apparatus according to claim 114, wherein the combiner further includes:
5 an implementation of a first bitwise binary function operative on said first segment and said encoded value to produce a first result;
an implementation of a second bitwise binary function operative on said second segment and said encoded value to produce a second result; and
a combiner to combine said first result and said second result to produce said
10 modified master key.

116. An apparatus according to claim 115, wherein the combiner includes a concatenator to concatenate said first result and said second result to produce said modified master key.

117. An apparatus according to claim 112, wherein the bit selector is operative to select a set of least significant bits from said hash value as said derivative key.

118. An apparatus according to claim 112, wherein the combiner is operative to
20 said master key and a counter to produce said modified master key.

119. A data security device, comprising:
a key deriver, including:

an input port to receive a master key;
25 a divider to divide said master key into a first segment and a second segment;
a repeater to repeat a value to form an encoded value as a longer bit pattern;
an implementation of a first bitwise binary function operative on said first segment and said encoded value to produce a first result;
an implementation of a second bitwise binary function operative on said
30 second segment and said encoded value to produce a second result;
a combiner to combine said first result, said second result, and said encoded value to produce said modified master key;
a hasher to hash said modified master key into a hash value; and
a bit selector to select a set of bits from said result as a derivative key; and

an encrypter to encrypt data using said derivative key.

120. A data security device according to claim 119, further comprising a data transformer.

5

121. A data security device according to claim 120, wherein the data transformer is operative to transform an original master key into said master key.

122. A data security device according to claim 120, wherein the data transformer is
10 operative to transform said derivative key into a transformed derivative key.

123. A data security device according to claim 119, wherein the repeater is operative to repeat a counter to form said encoded value as a longer bit pattern.

15 124. A method for performing key derivation, comprising:
combining a master key with a value to produce a modified master key;
hashing the modified master key to produce a hash value; and
selecting a derivative key from bits in the hash value.

20 125. A method according to claim 124, further comprising repeating a bit pattern in the value to form a longer bit pattern.

126. A method according to claim 124, wherein combining a master key with a value includes computing a bitwise binary function using the master key and the value.

25

127. A method according to claim 124, wherein combining a master key with a value includes:

dividing the master key into a first segment and a second segment;

combining the first segment with the value to produce a first result;

30 combining the second segment with the value to produce a second result; and

combining the first result and the second result to produce the modified master key.

128. A method according to claim 127, wherein combining the first result and the second result includes concatenating the first result and the second result to produce the modified master key.

5 129. A method according to claim 124, wherein selecting a derivative key includes selecting the derivative key from a set of least significant bits from the hash value.

130. A method according to claim 124, wherein combining a master key with a value to produce a modified master key includes combining said master key with a counter to
10 produce said modified master key.

131. A method for encrypting a derivative key, comprising:
combining a master key with a value to produce a modified master key;
hashing the modified master key to produce a hash value;
15 selecting a derivative key from bits in the hash value; and
encrypting data using the derivative key.

132. A method according to claim 131, further comprising applying a data transformation to the master key before generating the derivative key.
20

133. A method according to claim 132, wherein applying a data transformation includes:
dividing the master key into a third segment and a fourth segment, each of the third segment and the fourth segment including at least one bit;
25 organizing the bits in the fourth segment into a number of groups, the number of groups equal to a number of bits in the third segment; each group having a same number of bits;
associating each of the groups with a bit in the third segment;
applying a permutation function to at least one of the groups according to the
30 associated bit in the third segment; and
constructing the transformed master key from the third segment and the permuted groups.

134. A method according to claim 132, wherein applying a data transformation includes:

dividing the master key into a third segment and a fourth segment, each of the third segment and the fourth segment including at least one bit;

5 computing a power as a function of the third segment, the power being relatively prime to a function of a predefined modulus;

 computing a result of raising a function of the fourth segment to the power;

 computing an exponential permutation as the result modulo the predefined modulus;

and

10 constructing the transformed master key from the third segment and the computed exponential permutation.

135. A method according to claim 131, further comprising applying a data transformation to the derivative key.

15

136. A method according to claim 135, wherein applying a data transformation includes:

dividing the master key into a third segment and a fourth segment, each of the third segment and the fourth segment including at least one bit;

20 organizing the bits in the fourth segment into a number of groups, the number of groups equal to a number of bits in the third segment; each group having a same number of bits;

 associating each of the groups with a bit in the third segment;

25 applying a permutation function to at least one of the groups according to the associated bit in the third segment; and

 constructing the transformed master key from the third segment and the permuted groups.

30 137. A method according to claim 135, wherein applying a data transformation includes:

dividing the master key into a third segment and a fourth segment, each of the third segment and the fourth segment including at least one bit;

 computing a power as a function of the third segment, the power being relatively prime to a function of a predefined modulus;

computing a result of raising a function of the fourth segment to the power;
computing an exponential permutation as the result modulo the predefined modulus;
and
constructing the transformed master key from the third segment and the computed
5 exponential permutation.

138. A method according to claim 131, further comprising encrypting the
derivative key.

10 139. A method according to claim 138, further comprising transmitting the
encrypted derivative key.

140. A method according to claim 131, further comprising transmitting the
encrypted data.

15 141. A method according to claim 131, wherein combining a master key with a
value includes:

dividing the master key into a first segment and a second segment;
combining the first segment with the value to produce a first result;
20 combining the second segment with the value to produce a second result; and
combining the first result and the second result to produce the modified master key.

142. A method according to claim 141, wherein combining the first result and the
second result includes concatenating the first result and the second result to produce the
25 modified master key.

143. A method according to claim 131, wherein combining a master key with a
value includes:
dividing the master key into a first segment and a second segment;
30 combining the first segment with the value to produce a first result; and
combining the first result with the second segment to produce the modified master
key.

144. A method according to claim 131, wherein combining a master key with a value to produce a modified master key includes combining said master key with a counter to produce a modified master key.