Tailored Key Encryption (TaKE)

Tailoring a key for a given pair of plaintext-ciphertext.

Gideon Samid
AGS Encryptions, Ltd, Israel

D&G Sciences – Virginia Technology Corporation
P.O.Box 1022, 6867 Elm St, Suite 200, McLean VA 22101-1022, USA
gideon@dgsciences.com
202-487-4434

Nominally the cryptographic process is defined as follows: Given a plaintext, M, find an encryption algorithm E and an associate key K, so that M can be transformed into ciphertext C. Where the object of the transformation is to insure that without knowledge of K (or a corresponding decryption key K'), it would be practically impossible to reverse the transformation.

Given these four elements: M, C, E, and K, one could pose a different question: Given M and C, find E and K such that C=E(M,K). Such E will be called Tailored Key Encryption (TaKE) since it is an encryption algorithm which makes it possible to tailor a key K for an (M,C) pair.

We explore the following questions:

- Is Tailored Key Cryptography of any interest?
- Does it exist?
- Can a useful instant be found?

Structural Contents

INTEREST

Cryptography is useful only in situations where a suspected message M has more than one plausible contents. In general, there are n possible messages: M1, M2, M3,...Mn, each of which would be a plausible candidate for M. These n messages must differ in a meaningful way, so that an unintended reader of M would have an incentive to pin-point the particular message. While there may be many more messages: M(n+1), M(n+2),.... which M could stand for, only the first n candidates exceed a plausibility threshold, and thus no further options should be considered.

For n>1, the existence of tailored key cryptography means that there exists an encryption algorithm E, and respective n keys: K1, K2, ...Kn such that for a given ciphertext C we have:

$$C = E(Mi,Ki)$$

for i=1,2,..n.

Which in turn means that if C was encrypted through E, it becomes impossible to use C for the purpose of differentiating among M1, M2,..,Mn. Hence C can be fully exposed without increasing the vulnerability of the secret message M. This immunity against the mathematical processing of C holds regardless of the invested computing power.

TaKE, therefore, if it exists, offers mathematical security. But it goes even further: TaKE offers deniability.

In the section below we explain deniability, and its relationship with mathematical security.

DENIABILITY

Deniability is defense against pressure to reveal the contents of a captured ciphertext. Suppose C is mathematically secure against cryptanalysis, and so a hostile agent can not cryptanalyze C and extract the corresponding plaintext M. This state of affairs is likely to drive the hostile party towards an alternative strategy of attacking the holders of M in ways outside the realm of cryptanalysis. That attack is aimed at securing a leverage of extortion which would make it attractive for the holders of M to buy relief by trading the identity of M. In general the holders of M are the writer of M (and C), and the reader of the same. It is sufficient that one of them succumbs to pressure and reveals M.

If the respective key space includes only a single key, K, which decrypts C to a plausible M, then the coerced holder of M can not “cheat,” and must submit the real K. Any other K will not decrypt C to a plausible message M.

If the employed encryption was a TaKE type, then there are at least two plausible messages M1 and M2 with corresponding keys K1 and K2, and the coerced holder of M will be able to choose between K1 and K2, both will look reasonable to the extortionist. We say that TaKE encryption will allow its user to deny sending or receiving the real message as the one that was communicated in C, and unshakeably claim a decoy one.

Deniability as such will serve to discourage the would-be extortionist. There is little point in exerting pressure, if it is not likely to produce the target M. Worse: it may convince the extortionist that the decoy message is the true one.

DENIABILITY AND MATHEMATICAL SECURITY

For any finite size key space mathematical security is only achievable through deniability. Otherwise a cryptanalyst will work his way through all the possible keys and find what he is after. However, for a practically infinite number of keys one could potentially establish mathematical security without deniability.

EXISTENCE

Qualified, or restricted instances of TaKE do exist. Most notably, OTP (One Time Pad). Given C and M, two bit strings of equal length, OTP encryption will guarantee the existence of a key K such that:

$$C=E(M,K)$$

(E=Eotp). However, OTP is very inconvenient. What about today's popular and practical encryption engines? It is well known that none of them is TaKE compliant. For DES, RSA and their prevailing extensions, enhancements and variants, it is exceedingly difficult to generally find two distinct coherent messages M1 and M2 such that two respective keys K1 and K2 will satisfy:

$$C= E(M1, K1)= E(M2,K2)$$

for a given C, even for a choice C. And to find n>2 such messages (and n keys) is practically impossible. This is so regardless of contextual plausibility, which would further restrict the probability of finding valid TaKE options. It is this absence of TaKE which is the basis for cryptanalysis of these popular systems. One searches the key space looking for the first key that would yield a coherent M. No further search is necessary, since the chances that another key will yield a different plausible message are virtually zero.

OTP features TaKE with a restriction that the size of M and C is the same. This restriction becomes increasingly difficult as the size of C becomes larger. Consider a long message M which is OTP processed to yield an equally large ciphertext, C. OTP will allow one to match any other equal size message M' to C. However, the longer M', the more difficult is it to construct a highly plausible M'.

If M' could be of an arbitrary smaller size that difficulty diminishes. It is generally easy to construct short decoy messages, which are highly plausible. This reality poses the question: Is there an encryption system which will match a small as desired plaintext M to a fixed size C?

MATCHING SMALL M WITH LARGE C

We describe below a variation on the one-time pad (OTPv), designed to prove the existence of a TaKE algorithm where M < C or M << C.

In OTP the i-th bit of C, (ci), and the i-th bit of M, (mi), are matched through the i-th bit of the key K, (ki), such that:

$$ci = f(mi,ki);\ mi = f(ci,ki)\ ;\ ki = f(ci, mi)$$

where f is the "exclusive-or" relationship.

We may replace the i-th bit of K with a p bits string, and devise functions f', f" such that:

$$ci = f'(mi,kpi);\ mi = f'(ci,kpi)\ ;\ kpi = f"(ci, mi)$$

where kpi is the i-th group of p strings in K which becomes pl bits long (where l is the length of C and of M).

f' may be any function that would reduce the p- key bits into a single bit (which can be exclusive-or-ed as in f above). Such may be to focus on the first, second, ... or p-th bit, or perhaps to compute a checksum of the p bits.

f" may be any arbitrary selection of the p bits of k, as long as the selection would be compliant with the preceding f' relationships.

We have thus shown that it is possible to build a variant of OTP where the key is arbitrarily longer than the cipher C. Now, because of the symmetry between C,M and K it is possible to refer to K as the ciphertext, and the ciphertext will then take up the role of the key. (K <--> C).

We have thus a ciphertext C which may be matched to a plaintext M which is p times smaller. Where p may be 2, 3,....Lc (where Lc is the bit size of C).

And if the value of p is determined by a special bit string in K, then it is possible to match a ciphertext C with any plausible message M, no matter how short. And as indicated above, there are always plenty of plausible short messages.

A USEFUL TaKE SYSTEM

We construct a useful TaKE by revisiting the cryptographic key. We then construct a random-size, non-linear array, and present it as a working TaKE key.

KEY RE-DEFINED

Historically, the cryptographic key was the small piece of information which a user would commit to memory. As cryptography expanded beyond its exotic spy era, the smallness of the key was not critical any longer, but it remained a goal on account of computational load. Bruce Schneier has offered a modern definition of the cryptograpic key: a small (in size) secret which is used to protect a larger secret. Using Kerchoff principle one could chart another definition: the cryptographic key is the total of what needs to be replaced if a system is compromised, and is to be restored into its full strength. According to this definition one would favor a key with a great deal of variability, high entropy, large key space, so that the recovery act would be hard to follow. One could also offer a communication oriented definition. In a communication environment one encounters two types of channels: relatively secure, and relatively insecure. The secure channel is less available than the insecure one. The sender tries to cram into the secure channel as much of the message as possible, leaving as little as possible for the insecure channel. We may denote the part of the message that is communicated via the secure channel as “the key”. Accordingly, a mathematically secure system would be one where so little is communicated via the insecure channel that it is insufficient for reconstruction of the full message. A key, thus, should be constructed so that it may serve as a vehicle for as-needed large quantity of information. This requirement is not well served by a fixed-size binary string.

RANDOM SIZE NON-LINEAR ARRAY: A Novel Key Structure

Consider two alphabets: X1, X2, X3, ...Xn and Y1, Y2, Y3, ...Ym

Now consider a graph where some vertices are interconnected through edges. The vertices are marked by letters of the X alphabet, and the edges are marked by the Y alphabet. The only restriction on this design is the following: no two edges which share a vertex can be marked by the same symbol.

A graph so constructed may offer a mapping function from a string written in the X alphabet to a string written in the Y alphabet.

We denote that graph as a guide-sequence map, or gs-map.

Consider a string: Xi,Xj,Xk,.....Xg, to be called 'guide'.

Now construct a gs-map so that it contains vertices marked as Xi, Xj, Xk,...., and those vertices are also connected (by edges) in the same order. This construction will allow us to refer to the guide as a 'tour guide' which defines how a traveler is to chart a path on the map. Starting at Xi, hopping to Xj, on to Xk, etc. This path can also be defined by identifying the edges along its route. Once Xi is named, the second symbol could be the Y identifier of the edge that leads from Xi to Xj. Since there are no two edges which end at Xi and are of the same name, then the naming of the edge fully defines the next step. Hence, the guide can be written as the following sequence:

Xi: Yt, Yr, Yw,.....

Anyone in possession of the gs-map will readily reconstruct the guide from the Y sequence. Anyone without the knowledge of the gs-map will be unable to do the same because it is easy to map the sequence into any equal size guide of choice say: g'=X'i,X'j,X'k....X'p. Simply mark p vertices with the desired guide and connect them with edges marked according to the given Y sequence.

In other words, the gs-map has such variability that it may be readily constructed to match any two equal size strings. The gs-map may be of a much larger size than is flushed out in the guide-sequence mapping, and that part can be used for subsequent communication.

This procedure may work for mapping one alphabet to the same (X=Y). By referring to the gs-map as a cryptographic key, we have now constructed a setting, where any two equal size strings (one ciphertext, one plaintext), may be matched with a connecting key. Hence this procedure qualifies as a tailored key encryption (TaKE).

We may now regard the question of usefulness.

MAKING THE TaKE KEY USEFUL

The array key described above appears worse that OTP with regard to key size. This dilemma may be handled in two ways.

- Exchanging Key and Ciphertext
- Contaminated Repeat Patterns

More fundamentally, the gs-map (the array key) better be constructed in a way that would guarantee: Control of Cryptanalytic Difficulty and General Case Usability. These aspects are guaranteed through the Repetition Procedure.

All the above are addressed below.

THE REPETITION PROCEDURE

This procedure is designed to address the problem of general case usability and control of cryptanalytic difficulty.

These two problems are described below, followed by a description of the repetition procedure, and an elaboration on how it solves its target problems.

THE GENERAL CASE USABILITY

As for any other encryption scheme, the described TaKE should satisfy the following condition: the key should be so constructed that it would be able to map any given X-string to its corresponding Y string.

While the key can be as large as desired, it must be such that it is not about to encounter an X string that it can not encrypt. This would happen if a certain vertex is not edge-connected to a vertex marked by the next letter in the X string (the guide).

CONTROL OF CRYPTANALYTIC DIFFICULTY

Unlike the nominal case where cryptanalytic difficulty is measured by the effort to flush out the encrypted message, M; in a TaKE environment, that difficulty is measured by the number of highly plausible messages M1, M2, M3, ...Mn which would serve as high likelihood candidates for the

captured ciphertext C.

The designer of a TaKE system would wish to increase the number of M candidates to bolster its security.

While in theory one could argue that even for two candidates the system is secure, the reality is that external circumstances constantly change the plausibility of messages, and as time goes by, formerly plausible messages become less so. And hence the more candidates to begin with, the better.

DESCRIBING THE REPETITION PROCEDURE

This definition is based on the following notions:

- repetition free string
- induced repetition string
- zone
- the full-access condition

Based on these notions the repetition procedure. will subsequently be defined.

REPETITION-FREE STRING

Given any string written with the X (X1, X2, X3, ...Xn) alphabet, it is possible to render it repetition-free by incorporating an additional symbol (letter), say X0.

A repetition free string is one where there are no two identical adjacent letters.

To render a general string written in the X alphabet repetition-free, one would introduce the X0 letter between any two identical adjacent letters. The result will be a longer string, written through up to (n+1) symbols, but a string without any repetition.

REPETITION-INDUCED STRINGS

A repetition-free string can now be induced with any combination of repetition which will increase its size at will. However each of those repetition-induced strings will readily and unequivocally collapse into its repetition-free original. This would be done by replacing any block of adjacent identical letters with a single occurrence of this letter. There is no confusion about it, since the repetition-free string had no repetitions at all.

Say then that the original X-written string can be transformed into any larger size repetition-induced string.

While it is obvious how to collapse a repetition induced string to its repetition free mode, when written in the X alphabet, the situation is different when encrypted into the Y alphabet. Here without the gs-map, the same process of "collapsing the string" is impossible.

THE CONCEPT OF ZONE

A zone is a group of vertices marked by the same letter (of the X alphabet). The zone vertices are interconnected so that from every vertex one could step over to any other zone vertex, without stepping outside the zone.

THE FULL-ACCESS ATTRIBUTE

A gs-map will be considered a full-access map if for every zone therein there exists at least one edge which connects a vertex in that zone with at least one vertex of each of the remaining letters in the X alphabet.

DEFINING THE REPETITION PROCEDURE

We define encryption and decryption based on the following:

A plaintext M written in the X alphabet (X1, X2, X3, ...Xn) is to be encrypted into a ciphertext C written in the Y alphabet (Y1, Y2, Y3, ...Ym).

The encryption and decryption take place with the aid of a key in the form of a random-size full-access gs-map marked with an agreed upon starting vertex.

ENCRYPTION: Follow these steps:

- 1. Modify M to its corresponding non-repetition string, M'.
- 2. Induce repetition into M' so that it can be mapped through the gs-map into C.

C is written in Y (Y1, Y2, Y3, ...Ym).

The full-access attribute of the gs-map will insure that M' can be expanded through induced repetition so that for any possible M there would be an M' which can be defined as a sequence of edges on the gs-map.

DECRYPTION: Follow these steps:

- 1. Trace C on the gs-map and re-create M'
- 2. Collapse M' into M

ELABORATING ON THE REPETITION PROCEDURE

The repetition procedure allows for every possible string written in the X alphabet to be mapped by any full-access gs-map, regardless of size. This means that one could establish a small size key (gs-map) and use it indefinitely for any and all X-written messages.

The same procedure also insures that regardless of how large C becomes it will be a good match for a large number of plausible plaintexts. It makes sense to hide a sensitive piece of data through a great deal of induced repetition. Repetition insures that an L-symbols long C can be matched with instances of M of lengths: L, L-1, L-2,.....,3,2,1. Among which there should be sufficient plausible ones.

In addition, repetition offers interesting features which are absent in the prevailing cryptographies. The extra data that is added to M may be used for various purposes, like:

- adding verifications signals.
- annotation of the main message
- hiding a deeper message

By processing the extra data as another TaKE, these features can be nested at will.

EXCHANGING KEY-CIPHERTEXT

In a TaKE environment the natural process (M,K)->C, is complemented with: (M,C) --> K, which allows for (C,K) --> M (deniability). In turn this means that any two of the (M,C,K) triumvirate do not 'nail' the other. Or, in other words, no two items carry enough information to determine the third. This feature creates a situation where C and K can be exchanged. C can be communicated in the more secure channel, while K will be sent over in the less secure one. This exchange allows one to send the large body of data securely, and the smaller body in a more risky way.

CONTAMINATED REPEAT PATTERN

One could build a large as desired gs-map by combining smaller maps. If G1, G2, G3, ...Gk are k full-access gs-maps, then they can be combined into:

$$Go = G1 + G2 + G3 + ... \ Gk$$

which will also be a full-access map.

Combining two maps happens through connecting edges. One edge connecting one vertex of each map will suffice.

By allowing:

$$G1 = G2 = G3 = ... Gk$$

One may build an infinite gs-map. Alternatively a k distinct full-access gs-maps may be combined in any desired way into any number of repeat pattern configurations.

These repeat patterns may be 'contaminated' by any number of map slices which may be interjected anywhere. Such slices may be a full-access map, of any size, or they may be an enlargement of one or more zones (which will maintain the full-access property for the entire map). Since the described TaKE is a stream cipher, any such contamination will set off the definition of the cipher for its remaining length.

These are all means of creating a de-facto infinite key, without actually communicating an infinite amount of detail.

PRODUCT CIPHERS

The benefit of TaKE cryptography can be preserved to a great extent by combing it with any trusted cryptography of choice. The product cipher will offer the old cryptanalysis obstacles fostered by the deniability benefit.

The combination will have the TaKE cryptography preceding the OLD one.

The original plaintext M will be TaKE encrypted into C' which in turn will be OLD-encrypted into C. The total cryptographic distance (M<-->C) will be composed of the TaKE distance (M <--> C'), and the OLD distance (C'<-->C).

SUMMARY NOTES

The specter of Tailored Key Encryption (TaKE) should tickle the imagination of every cryptography enthusiast. We have outlined the fascinating potential, the operational desirability, the existence,

and the practicality of an environment where one could easily compute a cryptographic key that would encrypt a plaintext of choice into a given ciphertext. This feature amounts to plausible deniability. Deniability is not only a built-in mathematical security, it also offers defense against threats, coercion and pressure to reveal sensitive and private information. And as such it discourages such pressure from ever being applied.