ISSN: 3066-1730

Research Article

Journal of Electrical and Computational Innovations

Non-Trivial Ciphertexts: Decryption Variety, Contents Discrimination

Gideon Samid*

Electrical, Computer and System Engineering, Computer and Data Sciences, Case Western Reserve University, Cleveland, OH, United States

***Corresponding Author**
Gideon Samid, Electrical, Computer and System Engineering, Computer and Data Sciences, Case Western Reserve University, United States.

Submitted: 2025, Aug 20; **Accepted:** 2025, Sep 23; **Published:** 2025, Oct 08

Citation: Samid, G. (2025). Non-Trivial Ciphertexts: Decryption Variety, Contents Discrimination. *J Electr Comput Innov, 2*(2), 01-09.

Abstract

A trivial ciphertext is decrypted per all its bits to a corresponding, singular plaintext. A non-trivial ciphertext (NTC) is comprising decryption-proper bits as well as decryption unfitting bits (decoys) with a decryption discrimination key needed to identify each category. A non-trivial ciphertext may also decrypt to different plaintext options, determined by choice of decryption keys. NTC offer important advantages: giving ad-hoc power to buy extra security paid for with an inflated ciphertext, prospectively changing the strategic balance between cryptography and cryptanalysis. NTC involve plaintext alphabet that includes a 'plaintext empty letter' (PEL). Using a particular decryption key, certain ciphertext bits may decrypt to the PEL, and hence have no impact on the decrypted plaintext message constructed from the other bits. Using a different key, different ciphertext bits will decrypt to the PEL while a different collection of the remaining bits will decrypt to a different plaintext message. Thereby one achieves contents discrimination and decryption variety. The number of decoy bits is open-ended and is unilaterally controlled by the transmitter who thus determines the level of projected security -- asymptotically up to Vernam's perfection.

Keywords: Randomness, Equivocation, Vernam Cipher, Mathematical Secrecy, Cryptography-Cryptanalysis Balance

1. Introduction

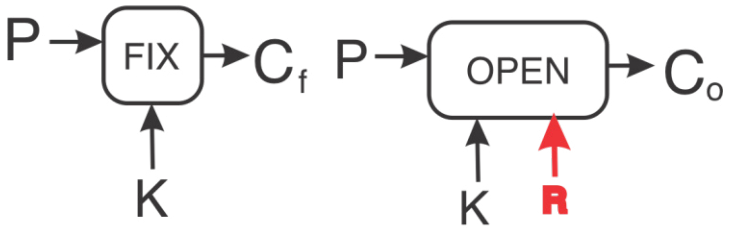
Presenting two manifestations of non-trivial ciphertext. One amounts to mixing the contents-bearing bits of a proper ciphertext with contents-devoid bits which blend into the contents-bearing bits. The other amounts to extending the trivial ciphertext to a ciphertext that decrypts to different plaintext messages upon use of different decryption keys.

These two manifestations are distinct but related. They both lead to a larger ciphertext. Such 'inflation' is the main price paid for the benefits of the non-trivial option.

In both cases the transmitter assumes control over the level of security projected by the ciphertext. Such security may be extended asymptotically to mathematical grade (Vernam cipher equivalent), thereby potentially changing the strategic balance between cryptography and cryptanalysis.

Discussing the two manifestations of non-triviality:
It is of a clear advantage to mix content-bearing ciphertext bits with content-devoid bits such that all the bits will carry a probability for being contents-bearing as viewed by an attacker who does not possess a discrimination key to separate the two categories of bits. At the very least, the more content-devoid bits that are being mixed in, the greater the cryptanalytic effort per same ciphertext. But more interestingly, the contents-devoid bits may per chance be interpreted as a different plaintext message, not the one that was encrypted into the contents-bearing bits. This might trap the attacker to a false conclusion.

Security at any Desired Level Through Open Ended Unilateral Randomness



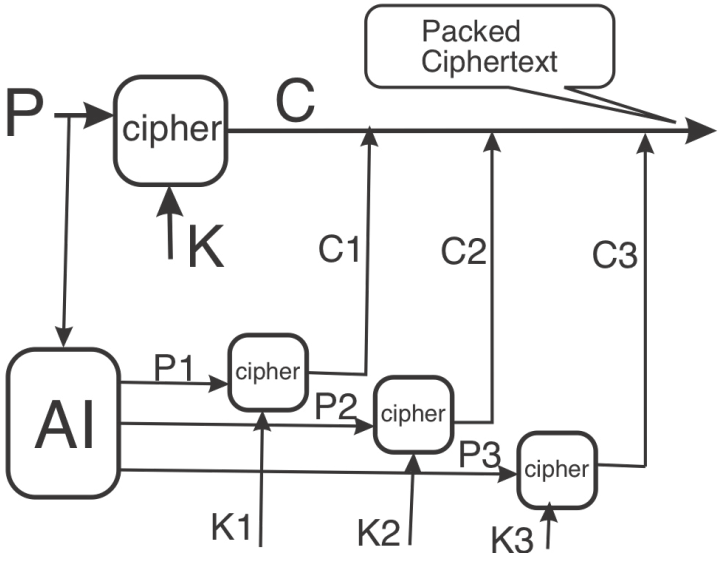
Plaintext P --> Ciphertext C using limited shared randomness key K

Plaintext P --> Ciphertext C using limited shared randomness key K plus open-ended ad-hoc randomness R

The process of adding contents-devoid bits to the contents-bearing bits of the ciphertext will be regarded as "ciphertext inflation". For inflation to be useful it is necessary that the attacker will be unable to distinguish between the two bit-categories while the intended recipient (recipient) will use a discrimination key to separate the two categories, and then decrypt the contents-bearing bits to the encrypted plaintext message.

It is also clear that it is of a definite advantage for one to pack along a given trivial ciphertext (the "payload" message) a collection of bits such that by applying to the packed ciphertext different decryption keys different plaintext messages (decoys) will be emerging. To the extent that the key used to extract the payload message is not identified in the packed ciphertext, and to the extent that the various plaintext messages emerging through different keys are all plausible per the case in point, it is impossible for however smart an attacker to decide which of the various decrypted messages is the payload and which are decoys.

Equivocating message P with math-equivalent AI generated decoy messages P1, P2, P3



A properly packed ciphertext thus, offers a terminal barrier ahead of complete cryptanalysis. Any attacker who derives their conclusions from only the ciphertext will have to list every plausible plaintext message as a likely option, no way to single out the right one.

We see then that non-trivial ciphertexts are very potent and powerful. It is worth investigating whether this concept is doable.

2. Cryptographic Framework

Let A^P be a plaintext alphabet comprising n letters: a_1, a_2, ... a_n. We define an "empty plaintext letter", a_0 that represents none of the n plaintext letters.

We associate each of the n plaintext letters with one or more

reference bit strings, referred to as "hooks", H
Letter a_i is associated with i' reference strings (hooks) h_{i1}, h_{i2}, ... h_{i}' for i=1,2,...n
There are |H| = **Σ** i'. for i=1,2,..n hooks. They are all different, no two hooks are identical. H is regarded as the encryption key.

We define a bit string as a 'clasp' , c, in relation to a 'hook h. A clasp may be fitting into a hook, where fitting is defined per a fitting algorithm, F.

We define a fitting algorithm F as an algorithm that takes in a hook for letter a_i, (h_{ij}) together with a clasp, c, and is issuing either a "0" (non-fit) or a "1" (fit):

$$\{0,1\} = F(h_{ij}, c)$$

We say that a clasp c_i points to a plaintext letter a_i, iff: for i=1,2,..n and for j=1,2,...i'
1 = $F(h_{ij}, c_i)$ for at least one j for j=1,2,...i'
and:
0 = $F(h_{kl}, c_i)$ for all k=1,2,.(i-1), (i+1),..n and all l = 1,2,..k'

As defined it is necessary to account for the full bit contents of all the hooks in order to determine if a particular clasp fits (points) into a particular hook.

These H hooks evaluated with the fitting algorithm, F, will be regarded as the 'hooking terms' or say 'pointing terms' (Alternatively 'the hooking test' or the 'pointing test') . A bit string c_i that satisfies the pointing terms vis-á-vis letter a_i, will be regarded as a ciphertext letter corresponding to plaintext letter a_i .

Any string c (clasp) that points to no letter in the plaintext alphabet, will be regarded as pointing to the empty letter, a_0 (PEL).

Any string c (clasp) that points to two or more letters in the plaintext alphabet will also be regarded as pointing to the empty letter a_0 (PEL).

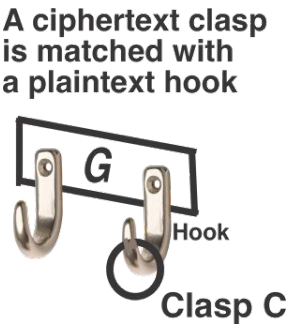
Accordingly, any sequence of clasps t_1, t_2, ... t_q will point each by order to one of the (n+1) letters a_0, a_1, ... a_n. These pointed-to ordered plaintext letters will amount to a plaintext message M that corresponds to the q clasps: T = t_1|| t_2 ||... t_q.

Anyone in possession of the hooks H will be able to evaluate the sequence of the q clasps, (T), to its corresponding plaintext message M. T will thus be regarded as an encrypted version of M.

The set of hooks, H, will be regarded as the symmetric cipher key which is used for encryption and decryption. The fitting algorithm F may be held as a secret, but will be considered here as publicly known.

One would require that given a hook h_i associated with letter a_i, it would be computationally easy to find a corresponding clasp c such that $F(h_i, c) = 1$. And given c and h_i it is required to be easy to evaluate F.

2.1. Inflation



A transmitter of message M comprising m plaintext letters. p_1, p_2, ... p_m will use a key comprising H hooks and readily list m clasps c_1, c_2, ... c_m such that for all i=1,2,...m:
$F(h_{ik}, c_i) = 1$. for at least one k where k=1,2,..i'
and:
$F(h_{jl}, c_i) = 0$ for all j ≠ i for all l = 1,2,....j'

There are two ways to find clasps: analytic, and randomized. One could set up an algorithm that would take in all the hooks and extract a clasp that qualifies as a valid pointer to a given plaintext letter.

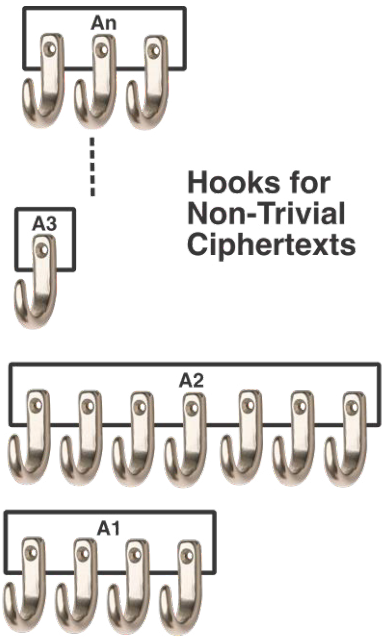
Alternatively, one would select a fitting algorithm F such that given a target hook, it will be easy to find a clasp such that the fitting algorithm will evaluate to 1. The selection of the fitting algorithm F will also be made such that the probability for an arbitrary clasp to evaluate as F=1 for other than the target hook will be sufficiently low. This will be regarded as the 'sticky probability', δ, (or alternatively the 'collision probability), namely the probability for a clasp to hook to an arbitrary hook ("collision").

In order to find a ciphertext letter c_i (clasp) to point to plaintext letter p_i, one would identify a clasp c that evaluates to $F(h_{ij}, c) = 1$, for any given j=1,2,...i'. and then check it against all the hooks that are associated with the other (n-1) letters in the plaintext alphabet. The probability that c will point to any second letter (other than a_i) is Δ ≈ (|H|-i')*δ. where |H| is the count of all the hooks in H. By setting δ low enough, Δ will be low too. Note: by setting |H| to be a low number, Δ will also be low. A low Δ implies that a random clasp pointing to plaintext letter a_i, is likely not to point to any other letter in A^p, and hence satisfy the pointing test.

In the event that the pointing test is not satisfied, the transmitter will randomly choose another clasp c' such that $F(h_{ij}, c') = 1$, and then evaluate c' against the other (|H|-i') hooks, and so repeat as necessary until the pointing test is satisfied and the resultant c^*_i clasp properly points to a_i.

Having found such proper clasps c_1, c_2, ... c_m for all the m letters in M, the transmitter would then transmit these m clasps (ciphertext letters) by order to the recipient, who would use the same key (same set H of hooks) to decrypt the m clasps to the corresponding

plaintext message M.



Randomized Hooks for each lettet of the plaintext alphabet

We may assume that the m clasps are concatenated into a combined message $C_M = c_1 \| c_2 \| .... c_m$.
C_M will be the nominal ciphertext expressing the encrypted version of the plaintext message M.
The transmitter would send C_M to the recipient and accomplish a proper act of transfer of an encrypted message. However, the transmitter might choose to pile on some g 'decoy' strings (false clasps) that do not pass the pointing test. These decoy strings may be comprised from clasps that were selected to pass the pointing test but failed to do so. They may also be selected from randomized strings that owing to the low value of δ are most likely to evaluate F=0 for all the hooks.

The so identified g decoy strings d_1, d_2, ... d_g can be injected anywhere in the original sequence $c_1 \| c_2 \|... \| c_m$, creating an inflated ciphertext comprising q = m + g strings.

No matter how many decoy strings are used, the recipient will successfully discard them all (point them to the empty plaintext letter a_0). Any hacker, not equipped with the key H will regard each of the q strings as having a certain probability for pointing to a plaintext letter.

As indicated, at the very list the g decoy strings will steal the hacker's attention. What is more, there is a chance that the inflated ciphertext will lend itself to interpretation where a different message has been encrypted, not M.

An inflated ciphertext will readily lend itself to be a tool to hide meta communication data.

3. Packing

Let H be one key (a set of hooks for all the letters of the plaintext alphabet). Let H' be another key such that no hook, $h_{ik} \in H$ is the same as any hook $h_{jl} \in H'$ for i=1,2,...n for k=1,2, i', and for j = 1,2,...n and l =1,2,...j'

Let clasp c_i point to letter a_i when evaluated through key H. Namely clasp c_i passed the pointing test vis-á-vis a_i. Let us further require that:

$$F(h'_{jk}, c_i) = 0$$

for j=1,2,...n. for k=1,2,..j'. Namely the clasp c_i does not point to any hook that connects to any plaintext letter, through the key H'.

The same is required in reverse, namely:
Let clasp c'_i point to letter a_i when evaluated through key H'. Namely clasp c'_i passed the pointing test vis-á-vis a_i. Let us further require that:

$$F(h_{jk}, c'_i) = 0$$

for j=1,2,...n. for k=1,2,..j'. Namely the clasp c'_i does not point to any hook that connects to any plaintext letter, through the key H.

We regard these |H|+|H'| conditions as the H-H' '*separation condition*'.

Let M be a plaintext message comprising m plaintext letters which is being encrypted using key H, generating ciphertext sequence C comprising q clasps that point to letters in the plaintext alphabet A^p to recreate M.

Let all the clasps that comprise C satisfy the separation condition. C will then evaluate (decrypt) to M by a recipient using key H. However, a recipient using key H' will evaluate C to a sequence of m empty letters,

$$M = Dec(C, H); PEL = Dec(C, H')$$

Let M' be a plaintext message comprising m' plaintext letters which is being encrypted using key H', generating ciphertext sequence C' comprising q' clasps that point to letters in the plaintext alphabet A^p to recreate M'.

Let all the clasps that comprise C' satisfy the separation condition vis-á-vis H. Namely:

$$F(h_{jk}, c_i) = 0$$

for i=1,2,..m'. j=1,2,...n, k = 1,2,...j'

C' will then evaluate (decrypt) to M' by a recipient using key H', but will evaluate to a sequence of m' empty letters when decrypted using the key H.

$$M' = Dec(C', H'); PEL = Dec(C', H)$$

We now create a 'packed ciphertext' C_{pack} by combining the C letters with the C' letters in any arbitrary way while preserving the internal order of C and C'. Namely for any i > j clasp c_i will follow clasp c_j in the packed ciphertext and clasp c'_i will follow clasp c'j in the packed ciphertext.

When the user of key H will apply it to decrypt $C_{pack,}$ the result will be message M. And when the same ciphertext pack, C_{pack} will be read by a user of key H' the result will be message M'.

As described a ciphertext pack C_{pack} is decrypted to two different plaintext messages each associated with a dedicated key H, or H'.

C_{pack} does not include the key that is used to decrypt it and thus anyone attacking C_{pack} will at most identify M and M' but will not be able to determine which message (M, or M') is the one decrypted by the intended user.

The transmitter of M may be sharing the key H with its corresponding recipient, and not share H'. The transmitter then will construct the C_{pack} with H (shared with the recipient) and with H' (not shared with the recipient). The recipient will interpret C_{pack} as M and will not realize that C_{pack} contains an alternative message M'.

Much as we described two independent messages packed into C_{pack}, we can envision any number t > 1 of plaintext messages packed into C_{pack}, each extracted with a dedicated key. An omnipotent attacker will at most identify the t messages packed into C_{pack}, but will not be able to point to the one that was communicated to the intended recipient. This is a built-in equivocation barrier changing the face of cryptanalysis.

The larger the number of keys used (the higher t) the more demanding is the separation condition. It will have to apply vis-á-vis all the hooks in the various alphabets. Namely every two keys in the set of t keys will have to comply with the separation condition.

4. Pattern Devoid Cryptography

The class of ciphers most fitting to practice non-trivial ciphertext is the pattern devoid type [1]. A pattern devoid cipher is minimizing arbitrary selections and maximizing random choices. Security is projected through randomness rather than through mathematical complexity. Mathematical complexity may disappoint its designer -- a shortcut may be found by someone smarter than the complexity builder. Randomness from a true source is rather immunized against any surprise pattern detection. It is pattern devoid.

The key in a non-trivial ciphertexts (NTC) is of random size. The number of hooks per each plaintext letter is unknown. Depending on the cipher the size of each hook may be unknown too. Given a hook it is easy to find a fitting clasp. Once a clasp is found it must be checked against all other hooks to ensure there is no double-pointing. Since the hooks are randomized there is no analytic way to check for double pointing except to actually check a given clasp vis-á-vis all the hooks that point to a different letter. The parties control the probability for a randomized clasp to show double pointing. If the probability is high, it might take the transmitter several rounds to find a clasp that hooks into no second hook (collision free).

5. Operation

Non trivial ciphers may be operated in a trivial mode. That is, no decoys are being used and no packing of alternative messages is happening. We describe first the most basic application: every letter of the plaintext alphabet has only one hook, and the fitting algorithm is the simplest possible, equality. A hook, h combined with a clasp c, will evaluate to a fitting value F=1 iff c=h.

Accordingly, one could use Base64 as the plaintext alphabet and assign to each of the 64 letters a hook comprised of 6 bits. The fitting clasp will be identical to the hook and of same length (6 bits). So, described we have defined a simple permutation cipher where each of the base64 letters is represented by another letter from the same alphabet. There are established ways to crack such a fixed permutation cipher and they would work here.

This basic operation can be made a bit more complicated by using 7 bits hooks. This will allow one to use 128 distinct hooks in the key H. The extra 64 hooks will be spread randomly among the original base64 letters. So, say the letter g will be represented by 3 hooks, and the letter e with 4 hooks etc. Since this extra hooks allocation is randomized, there is no analytic cryptanalysis possible, only brute force.

More latitude, more security is projected for larger hooks, say 8, 9, 10,... bits each. The ciphertext will be extended, true, but security will be well served.

More security will be projected by switching the fitting algorithm from equality to a different test. Various pattern devoid ciphers offer interesting and powerful fitting algorithms. One example is the polar lattice cipher where a hook is defined through starting point and ending point of a trail on a secret polar structure and the clasp is the bit sequence of the trail that leads from the starting point to the end point. A simpler and older cipher is BitFlip [2].

Illustration: BitFlip

In the BitFlip cipher (basic version) the hook, h, and the clasp, c, are random bit strings of same size (x bits) [3]. The fitting algorithm F is F=1 if the Hamming distance is h* where the value of h* is engineered to handle the sticky probability challenge.

The number of strings of size x bits that have a Hamming distance of h* from the hook h is:

$$n(x,h^*) = x! / (h^*! (x-h^*)!)$$

The values of x and h* need to be adjusted such that there will be

many qualified clasps, on one hand, but that the probability for a random string of size x to be a clasp will be low.

Illustration: for x=100 and h*=20, we have: n(x,h) = 100! / (20! (80)!) = 7 * 10^{28}
which is a huge number. The chance for a random clasp of size 100 bits to fit into a hook is:
Pr = n(x,h)/2^{100} = 5%

If the plaintext alphabet is Base64, then the chance for a random string to be free of collision is: (1-0.05)64 = 3% only. By selecting h*=10 the numbers are recomputed:

$$n(x,h) = 100! / (10! (90)!) = 1.73 * 10^{13}$$

$$Pr = n(x,h)/2^{100} = 0.86 * 10^{-17}$$

With such small probability, there is no fear for collision.

So far, the non-trivial cipher was used trivially. No decoys added, no packing done. It is up to the transmitter to pepper the emerging ciphertext with as many decoys as desired, and pack it with as many alternative messages as called for by the circumstances.

One way to add decoys is by pushing forth clasps that failed the fitting test on account of sticking to a second letter. Say the transmitter selects a clasp c that fits to a hook associated with plaintext letter. a_1. Upon evaluating c with all the other hooks is turns out that F=1 computes for letter a_2. That clasp will fail the fitting test. The transmitter may choose to discard it and randomly look for another clasp, or the transmitter may choose to send c to the recipient counting on the recipient to conclude that this clasp is decoy since it fails the fitting test. Next the transmitter will look for another clasp.

The base mode may be upgraded by a unilateral decision of the transmitter to add decoys or to pack fake messages into ciphertext. Neither decision requires pre-coordination with the recipient. The extent of adding decoys or packing is determined live by the transmitter based on on-going appraisal of the security requirements. It is noteworthy that the transmitter may push the security extensions to match up the mathematical security offered by Vernam cipher.

In particular, if the number of hooks equals the number of letters in the message then any plaintext message of same length can be matched to any given ciphertext, just like the case is with Vernam. The NTC allow also for some specific applications. Some are discussed below.

5.1. Meta Data Guard

Encrypted or otherwise, when parties are engaged in on-going communication they betray much of their activities and intentions merely through the pattern of the exchange -- known as meta data. NTC may be used to conceal this meta data. The idea is to establish a fixed rate of communication between a pair of parties where the default case is a "fakeload" -- decoy clasps that go from transmitter to recipient at a fixed rate. Should one party wish to communicate content to the other, this will be done through genuine payload clasps that would replace clasps from the fakeload. The intended recipient will readily distinguish between the payload and the fakeload, but the eavesdropper will see an indistinguishable flow of clasps being unaware if any or all of the passing clasps are genuine or not.

In the case of two parties each party will send the other a fixed data stream that is intensive enough to fit the communication intensity when actual payload is passing through. In the case where a large number n of parties wish to hide their meta data they might have 0.5n(n-1) paired data flows, or they may be configured around a communication center (hub) where each party has a pair of data flows with that center. Communication between the parties will pass through that center. Such central architecture may be iterated. Some m centers (hubs) may be served by a certain super-center communicating on fixed rate with each center, and then again, if necessary, towards a higher hierarchy hub. Thereby minimizing the number of active data flows.

5.2. Administrative Security Levels

NTC may be used to handle administrative tasks where a project has several levels of exposures where the higher managers see more than lower ranked participants. In that case a given project data may be interpreted using a first key to a first exposed level, and by reading the same encrypted file with a second key a more sensitive data comes forth. This may be repeated for several levels. Any project reader will be given reading key for up to his or her level and no higher, so each project participant will read from the file only what they are supposed to see.

5.3. Entrapment

The second message in a two-message pack can use a smaller key, and yield first to a cryptanalyst. It may be set to be an entrapment message sending the hacker to activity that is harmful for him.

6. Security

One hundred and eight years ago the Vernam cipher was claimed as a patent by Gilbert S. Vernam. Twenty-five years later Claude Shannon has proven that this cipher is mathematically secure, and hence is not breachable by however a smart attacker and by however powerful computers [4]. Filing of this cipher could arguably be regarded as the most seminal moment in the history of cryptography. It appeared then that cryptography as a battle of wits between code writers and code breakers has been coming to its end.

It was not to be. The size of the required Vernam key and the necessary quality of randomness presented practical barriers for widespread use. Cryptography touched for a moment the golden standard of mathematical proof of efficacy, but hence on retreated to the unsavory standard of '*Absence of A Published Breach*' -- a standard it upholds till this very day. It is hard to argue in favor of this 'absence' standard given that a code breaker is well served

when the fact that they broke the code is not well known, and more so, is not even suspected. Therefore, code breakers are expected to deny having cracked a code they labored hard to crack. And they further argue that it is unlikely and indeed "impossible" to break the code they actually did break.

Over that worrisome state of affairs, one should see a bright ray of light coming from these non-trivial ciphertexts: they steer cryptography back towards the once-touched and then abandoned Vernam security. Claude Shannon has shown that Vernam security requires for the key to be as large as the encrypted message -- which undermines its utility. However, this requirement is linked to the Vernam high standard wherein every possible plaintext of the size of the ciphertext can be decrypted from the ciphertext, using a proper key. This is an excessive requirement. In any given practical situation, there are only a handful of plausible messages that could have reasonably been suspected to be the encrypted ones. All that is practically needed is for these plausible messages to be decryptable from a given ciphertext, and security will be practically equivalent to Vernam.

Using the NTC all the plausible messages representing a given circumstance may be packed into the packed ciphertext. Such packing will construct a ciphertext with effective security. If fewer than all the plausible messages are packed into the ciphertext then security will be less than Vernam but nonetheless presenting a terminal equivocation which the cryptanalyst can negotiate no further.

NTC operates with a key of unknown size. If the key is made to fit the size of the message then NTC is formally equivalent with Vernam. If the number of hooks is the number of letters in the encrypted message, then the encrypted messages can be decrypted into any plaintext of same or smaller size, using the same logic that applies for Vernam.

When NTC is applied with lavish use of decoys then the projected security is a matter of probability calculus which can be computed by the transmitter. Indeed, the NTC brings back the simple and powerful idea of math-proven or math-established security. One clear advantage here is that once a proof is articulated, the cipher is immediately ready for deployment. This is by contrast to the extended certification period required by the ciphers where security is based on 'absence of a published breach'.

In summary, non-trivial ciphertexts (NTC) return cryptography to the simple idea used by Vernam 108 years ago: math proven security.

7. TESTOR

Surprise threat is cyber reality, requiring quick and effective response. Such is the threat of (i) extra malicious attention to our encrypted messages, (e.g. harvesting for later breach), (ii) alarming suspicions or indications that our deployed cipher has been compromised, or (iii) an adversary is tracking our meta data to gain an advantage. The transmitter today can either assume the extra risk, or stop the transmission. Both reactions are unattractive. The NTC offers a remedy. Packed as a TESTOR Gear (Transmitter Empowered Security, Threat Oriented Response), providing a means for the transmitter to unilaterally increase the projected security of the transmission, without switching to a different cipher (which would be cumbersome), and without pre-coordination with the recipient (which may be complicated and slow) [5].

The transmitter may react to each of these threats by unilateral action that can be activated immediately without delay and can last for as long as the activation threat is present.

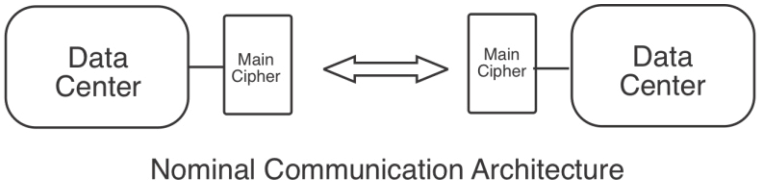
The advantage of the TESTOR is that it fits on top of any cyber security, it works in conjunction with any underlying cipher. It requires no removal of operating parts, no exchange, no shut down. The TESTOR security boost is open ended, in as much that security can be increased at will up to mathematical levels.

TESTOR operates over a continuum, it can be operated mildly, be pushed further, keep on increasing its impact for as much as its operator desires. The evident cost of the TESTOR operation is increased data handling and longer ciphertext. In today's 5G environment this is a very negotiable price.

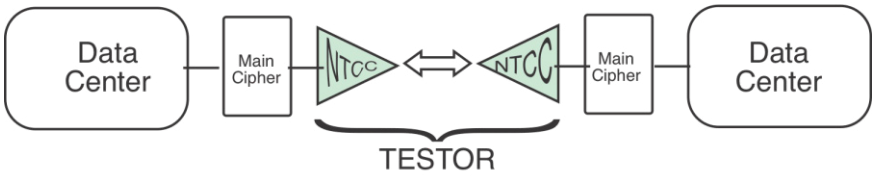
When the threat that invoked TESTOR is relaxing so does the TESTOR response. When the threat is over, so is the TESTOR operation. The data exchange then returns to where it was before the threat appeared and TESTOR sprang into action.

The transmitter of data is the party that is assumed to know the best the sensitivity of the data it transmits. It is the party that is close to the action, so it detects the threat fast, and since it can kick the TESTOR unilaterally the transmitter will operate without delay. If two parties run a conversation and a threat is common to both then both will activate the TESTOR for their transmission.

In the circumstances where a group of communicating parties are put under a sudden threat then all members of the group will go into TESTOR mode, each for their own transmission [6-30].



Nominal Communication Architecture



Transmitter Empowered, Threat Oriented Response

Non-Trivial Ciphertext Cipher (NTCC) offers instant Security Boost and Meta Data Control

TESTOR Architecture

References

1. Samid, G. (2023). Pattern Devoid Cryptography.
2. Samid, G. (2025). Polar Lattice Cryptography. *J Electr Comput Innov, 2*(2), 01-18.
3. Samid, G., & Popov, S. (2017). Bitflip: A randomness-rich cipher. *Cryptology ePrint Archive.*
4. Shannon, C. E. (1949). Communication theory of secrecy systems. *The Bell system technical journal, 28*(4), 656-715.
5. Transmitter Empowered Security, Threat Oriented Response.
6. Samid, G. (2023, July). AI Resistant (AIR) Cryptography. In 2023 *Congress in Computer Science, Computer Engineering, & Applied Computing (CSCE)* (pp. 145-149). IEEE.
7. Samid, G. (2023). Cryptographic Key Exchange: An Innovation Outlook. *Cryptology ePrint Archive.*
8. Samid, G. (2021). FAMILY KEY CRYPTOGRAPHY: Interchangeable Symmetric Keys; a Different Cryptographic Paradigm. *Cryptology ePrint Archive.*
9. Samid, G. (2024). Finite Key OTP Functionality: Ciphers That Hold Off Attackers Smarter Than Their Designers. *Cryptology ePrint Archive.*
10. Goldwasser, S., & Micali, S. (1984). Probabilistic Encryption. *Journal of Computer and System Sciences, 28*(2), 270-299.
11. Innovation Solutions Protocol.
12. Samid, G. (2025). Lifeboats on the Titanic Cryptography. *Cryptology ePrint Archive.*
13. NEPSAR: Secure Key Exchange.
14. Samid, G. (2018). Randomness as absence of symmetry. In *The 17th International Conference on Information & Knowledge Engineering (Ike'18) Las Vegas, NV.*
15. Samid, G. (2001, November). Re-dividing Complexity between Algorithms and Keys: (Key Scripts). *In International Conference on Cryptology in India* (pp. 330-338). Berlin, Heidelberg: Springer Berlin Heidelberg.
16. Samid, G. (2001, September). Anonymity management: A blue print for newfound privacy. In *The Second International Workshop on Information Security Applications (WISA 2001), Seoul, Korea.*
17. Samid, G. (2001, October). Encryption sticks (randomats). In *International Conference on Information and Communications Security* (pp. 150-154). Berlin, Heidelberg: Springer Berlin Heidelberg.
18. Samid, G. (2002, September). At-Will Intractability Up to Plaintext Equivocation Achieved via a Cryptographic Key Made As Small, or As Large As Desired-Without Computational Penalty. In 2002 *International Workshop on CRYPTOLOGY AND NETWORK SECURITY San Francisco, California, USA September.*
19. Samid, G. (2003, July). Intractability erosion: The everpresent threat for secure communication. In *The 7th World Multi-Conference on Systemics, Cybernetics and Informatics (SCI 2003).*
20. Samid, G. (2004). Denial Cryptography Based on Graph Theory. *U.S. Patent No. 6,823,068.* Washington, DC: U.S. Patent and Trademark Office.
21. Samid, G. (2008). The Unending Cyber War. DGS Vitco.
22. Samid, G. (2015). Equivoe-T: Transposition Equivocation Cryptography. *Cryptology ePrint Archive.*
23. Samid, G., & Wnek, G. E. (2019). The "Rock of Randomness":

a physical oracle for securing data off the digital grid. *MRS Communications, 9*(1), 67-76.
24. Samid, G. (2023). " Tesla Cryptography:" Powering Up Security with Other Than Mathematical Complexity. *Cryptology ePrint Archive.*
25. Samid, G. (2023). The Prospect of a New Cryptography: Extensive use of non-algorithmic randomness competes with mathematical complexity. *Cryptology ePrint Archive.*
26. Samid, G. (2025). Transmitting Secrets by Transmitting only Plaintext. *Cryptology ePrint Archive.*
27. Samid, G. (2023). The Prospect of a New Cryptography: Extensive use of non-algorithmic randomness competes with mathematical complexity. *Cryptology ePrint Archive.*
28. Burton Rosenberg University of Miami. (2020). THE VERNAM CIPHER AND PERFECT SECRECY.
29. Samid, G. (2019). SpaceFlip: Unbound Geometry Cryptography. *Cryptology ePrint Archive.*
30. Samid, G., & BitMint, L. L. C. (2018). Randomness rising. In *14th International Conference on Foundations of Computer Science (FCS'18: July 30–August 2018, Las Vegas, NV.*