Tidy: An Encryption Scheme With The Constant Presumption of a Compromising Intermediary Through a Non-private Medium

Amer Almadani
DRAFT
amer@live.ca

Abstract

The introduction of a set of processes whereby the encryption of data is based on a fundamental logic-gate(bitwise xor) that encompasses several chaining schemes within its main assumption. The principle operation relies on data deconstruction and therefore decryption by means of reconstruction given a public *Segment Identifier*. The method assumes a compromising intermediary *Eve* who has active access to both *Alice's* and *Bob's* raw packets. The scenario is further exacerbated by the assumption that the computational capabilities of *Eve* are superior to what is advertised. It is here that the introduction of **Tidy** as a set of operations can prove useful. The inclusion of *Rubbish* blocks to the encrypted data, the transmission of vicissitude versions of the encrypted data and the *Segment Identifier*, and the use of a *Formula Agreement* scheme to encrypt the *Segment Identifier* body; guarantees expensive participation by *Eve*.

1 Introduction

The imperatives of information security dictate a necessity of Confidentiality, Integrity, and Availability: *CIA Triad.* Despite efforts to utilize the agency of the triad, it has been shown that within its narrow or extended delineation[1], there contains a major flaw in its application [2]. Lundgren et al. have countered the traditional triad for a new definition of information security. They named it Appropriate Access: *AA*. Although it could be argued that the new definition is a protracted form of the triad, they support the new definition with compelling arguments. AA assumes relative relationships between information and parties; the example given in their paper makes a clear distinction between an information object, agents, and stakeholders. The scheme is contingent on the agent A having *appropriate access* to P as every part of object O, relative to stakeholder H. **Tidy** fulfills a re-imagined rearrangement of *AA* in that it assumes *Alice* and *Bob* as parts of H, *Eve* as an inevitable agent A, whereby A's appropriate access is limited to raw/encrypted packets P in encrypted object O, so that:

$$\top(O) \;\Leftrightarrow\; (\forall A \;\wedge \forall P) \;\in\; H$$

AA as a supportive criterion for **Tidy** aids the scheme in it having clear definitions of parties and information objects; a critical aspect of **Tidy's** main assumption. Since a clear distinction between secure and insecure states is a prerequisite for **Tidy** to be considered effective, basing it on the traditional triad would require redefining all elements of the triad to complement the various states and procedures of the scheme. It should be noted that *AA* is an information security definition and not a framework to base encryption standards upon.

AA has been utilized as a guideline for the main assumptive scheme since it has aligned perfectly with the problem's description. In the **Tidy** question, *Eve* is assumed to have direct access to all information objects flowing from and to *Alice* and *Bob* in the form of packets within the Public Operations Domain[2]. This is the supposition of *Eve* as a necessary agent A within *AA*. *Alice* and *Bob* are assumed to be the owners of the information objects in a secure state within the Private Operations Domain; the presumption of stakeholders H in the *AA* scheme. Earlier iterations of conjoining *AA* and the **Tidy** question had *Alice* as the sole stakeholder; *Eve* and *Bob* were agents, each with access to the information object relative to *Alice.* This was dropped since it was later posited that the true owner of the information object is *Bob*; being the intended recipient of *Alice.* The association of *AA* and the **Tidy** question is a crucial factor in the validity of the scheme. The proposed solution to the **Tidy** question presented in this paper relies on an abstracted take on Lundgren et al. work in a hypothetical example where *Alice*, *Bob*, and *Eve* are situationally and operationally aware of each other within the context presented in *AA*'s definition.

[1]Many iterations and frameworks of the CIA Triad exist, for example ISO/IEC 27001 and NIST SP 1800-26.
[2]A description of Private and Public Operations can be found in §3.1

2 The Problem

Tidy is based on an example where *Alice* and *Bob* are both under an active attack by *Eve*. *Eve*'s attack surface is initially limited to a singular attacking vector; an active-communication-session-hijack–**figure 1**, and inevitably an offline brute-force attack. The exact means[3] of achieving the active attack is irrelevant as long as the assumption of a compromised session between *Alice* and *Bob* is fulfilled. It also assumed that there is no apparent remedy to the attack vector, hence no mode in which the active attack can be circumvented without enforced encryption.

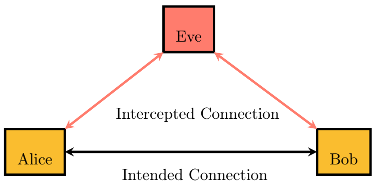

Figure 1: A generalized Man-in-The-Middle attack scheme

The example sees *Alice* sending a file to *Bob* whilst acknowledging the existence of a compromising intermediary whom will posses a copy of the file before its receipt by *Bob*. The computational capabilities of *Eve* are assumed to be vastly superior to what *Alice* has initially envisioned–*within the confines of commercially available technological advancements.* This compounds the problem of the **Tidy** question as follows:

$$\forall e \in C \implies C(e)\ as\ E \in P_{eve} \therefore \top(P_{eve}) \implies \bot(S(E))$$

for any element e in the active communication session C between *Alice* and *Bob*, *Eve* will have active and passive access as P_{eve} to $C(e)$ relative to stakeholder H given Lundgren et al. definition. Therefore, the assumption of a secure element $S(E)$ in negated.

Alice could opt for a symmetric encryption scheme. However, neither *Alice* nor *Bob* are willing to alleviate their communication standards in such a way as to rely on a singular cryptographic scheme[4], especially since all parties are aware of each other. The second element within the **Tidy** question that needs to be considered is a direct consequence of the lack of public key encryption as the exclusive mechanism. If the existence of *Eve* as a compromising intermediary is be to fully mediated without the dependency on public key cryptography alone, then a new mechanism needs to be introduced.

Since the **Tidy** question has thus far been inclusive of *Eve* as an agent, and since *Alice* and *Bob* have opted for a communication standard of their own; they need a means of authenticating each other before initiating communication and during the actual transmission of the information object. *Alice* and *Bob* cannot use symmetric encryption schemes due to the fear of interception and manipulation by *Eve*. However, they can utilize asymmetric encryption schemes, by means of public key encryption to facilitate the safe delivery of integral parts within the **Tidy** scheme. The use asymmetric public key encryption within the **Tidy** question could be limited to the safeguarding of the *Segment Identifier*[5] for example. To this effect, *Eve* will be limited to launching offline attacks on the intercepted information object. Yet, it is assumed here that neither *Alice* nor *Bob* can use asymmetric encryption algorithms due to key-pair reuse[6]. Although it might seem as a bold presumption–insecurity due to the inevitability of the key-pair reuse–it is assumed to be an inconvenient element and not a critical review of asymmetric encryption algorithms. The notion being that with the initiation of each communication session, a new key must be generated. In the

[3]Any or all combinations of techniques and Man-in-the-Middle attack types should be assumed; Sniffing, Packet Interception/Injection, (DNS, mDNS, ARP, https, IP) spoofing, BGB Misdirection, SSL hijacking/striping, Evil Twin/Rouge AP, etc...

[4]A detailed explanation can be found in §4; *On The Matter of Public Key Encryption.*

[5]§3.3

[6]The idea is that the security of the private key is at stake and therefore all secret messages between the parties can be decrypted. More on this in §4; *On the matter of Public Key Encryption.*

sense that the key from any given session cannot be used to decrypt the next encrypted information object.

The extent of the offline attack cannot be determined; although it has been assumed that *Eve*'s capabilities are limited to what is commercially available, it cannot be discerned who exactly is *Eve*? Knowing the identity of *Eve* could assist in modelling the scheme is such a way as to create specific deterrents to *Eve*'s ploy. It is therefore assumed hereafter, that *Eve* is a general malicious actor with access to the latest *commercial* advancements in software and hardware.

3 Tidy

Tidy aims to solve for the compounded expression where $\top(P_{eve}) \implies \bot(S(E))$ by introducing a scheme within a scheme whereby the negation is negated is such a way as to prove, that despite the compromised session, $S(E)$ is *True*: $\top(P_{eve}) \implies \top(S(E))$. The solution to this integral element is to allow for variability within the scheme given a set of agreed constants between *Alice* and *Bob*. The variability is a heterogeneous element within the scheme in the form of a *Formula Agreement*. The effectiveness of which relies on the agreement of three essential elements in person. Hence, for **Tidy** to function as intended and to solve for the compounded expression, *Alice* and *Bob* need to initiate their initial communication session in person. The elements that need to be agreed upon are an integer, a formula, and numerical substitution scheme.

Tidy is not a novel idea. Its main inspiration came from Gunpei Yokoi's "linear thinking with withered technology" [3]. Whereby the utilization of existing-proven cryptographic technologies is used to facilitate an auditable communication standard between semi-unauthenticated parties. It consists of three main elements within the Private Operations Domain[7].

- An XOR scheme for encrypting and decrypting data using the bitwise xor cypher; the scheme is dubbed *madXOR*. It includes multiple layers of data deconstruction and reconstruction. More on *madXOR* in §3.2.
- A *Segment Identifier* based in part on Lamport [1]. Among the many parameters that the *Segment Identifier* houses is the *madXOR* key. More in §3.3.
- A Formula Agreement scheme based on a Modulo operation as a means of shielding the *Segment Identifier*. More in §3.4.

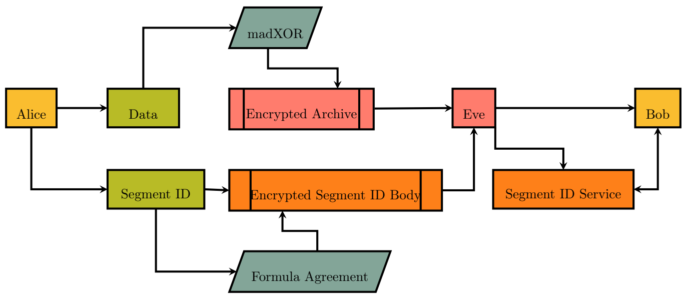

Figure 2: Tidy Operations Flow

Figure 2, describes the operation flow of **Tidy**. *Alice* prepares a file and generates the *Segment Identifier* to be sent to *Bob*; *shown in light green under data and Segment ID respectively.* The file will be encrypted through the *madXOR* scheme and the body of the *Segment Identifier* will utilize a Modulo operation through the Formula Agreement scheme; *dark green.* The encrypted file; *Encrypted Archive*, and the *Encrypted Segment Identifier* will pass through *Eve*. The encrypted file is assumed to reach *Bob* directly, whilst the *Segment Identifier* will be fetched by *Bob* after passing

[7]§3.1

through *Eve* from a public *Segment Identifier* Service. Note that secure states; Private Operations Domain, are coloured in shades of green(light and dark. Insecure states; Public Operations Domain, are coloured in red and orange.

If the assumption of **Tidy**–a set of operations for securing an information object in such a way as to circumvent *Eve*'s active and passive attack–is be considered safe[8], then the use of the aforementioned elements alone will not satisfy the assumption. *Eve* is still capable of offline attacks which might include various methods of breaking the xor cipher–REF or brute forcing the *Formula Agreement* scheme. It is here where a contingency on probability is considered. The *madXOR* and *Formula Agreement* schemes have been designed in such a way as to insure expensive participation by *Eve*. **Tidy** also contains an obfuscation element within the *Segment Identifier* operation, where a multitude of invalid *Segment Identifiers* are pushed with the correct one to the *Segment Identifier* Service as an obfuscation measure.

Given that **Eve** has gotten out of her way to mastermind an impressive attack, it should be assumed here, that *Eve* is devoted to her cause. This matter was noted in §2. A clear distinction between secure and insecure states must be prefaced before instigating the **Tidy** scheme. The prerequisites of **Tidy's** effectiveness rely on the complete segregation of Private and Public Operations. This is imperative since the example assumes active packet interception by means of an active session hijack or otherwise. All the while, *Alice* and *Bob* are to be assumed "safe" in their Private Operations. Private and Public Operations are generic terms for operations taking place on a local machine; *Alice's* secure state, and in the wild; vis-à-vis under active interception by *Eve*; an insecure state.

3.1 Public and Private Domains

Within the **Tidy** scheme, a clear distinction between Public and Private Operations is made. Public Operations are the actions taken with the assumption of active surveillance. Figure 3, shows all interactions between *Alice*, *Eve*, and *Bob* within the Public Operations domain. It should be noted that Public Operations are those which are assumed to be compromised by *Eve*. The various Public Operation undertaken by *Alice* and *Bob* are denoted as follows–*note that Public Operations are enclosed in a dashed box in figure 3 and are coloured in red*: The transmission of the encrypted data as *DT*. The *Segment Identifier* repository push and retrieval as *SP* and *SR* respectively. Although the retrieval of the encrypted data by *Bob* should have been denoted; it was dropped since *DT* entails its receipt by *Bob*.

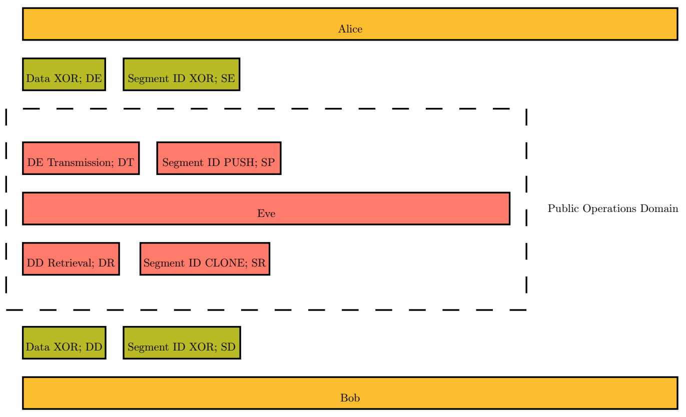

Figure 3: Interactions between *Alice*, *Eve*, and *Bob* within the Public Operations Domain

Private Operations are those which are assumed to be safe from interception; invulnerable local operations in a secure state. Figure 3 shows an abstract of the Private Operations conducted by *Alice* and *Bob*. The operations are denoted as follows–*note that Private Operations are coloured*

[8]Safe from interception

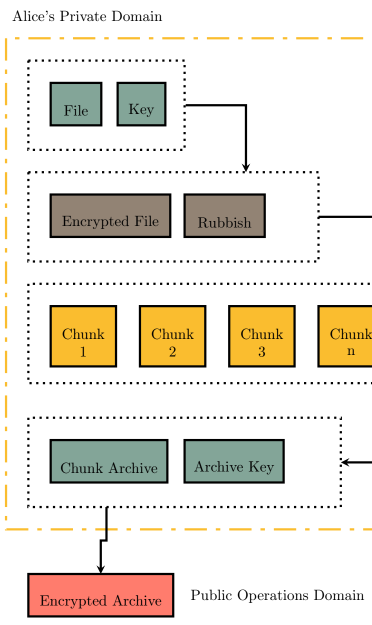

Figure 4: An abstracted flowchart of the *madEncrypt* scheme

in green: Data encryption by *Alice* as *DE*. The encryption of the *Segment Identifier* as *SE*. The decryption of the *Segment Identifier* and data decryption by *Bob* as *SR* and *DD* respectively. The aforementioned denotations are concerned with Public and Private Operations and should not be confused with the coming scheme denotations.

3.2 madXOR

madXOR describes a set of Private Operations within the principle scheme of **Tidy**. The encryption scheme(*madEncrypt*) is divided into three main sequential procedures:

1. An initial xor operation on the intended data.
2. Deconstruction of the encrypted data and the creation of chunk files.
3. A final xor operation on an archive folder containing the chunk files.

In Figure 4, *Alice*'s Private Operations are enclosed with a *yellow* dashed box. *Alice xor*'s the file with a generated pseudo-random key. *Rubbish* of size $1/n\ len(E_d)$ is added to the encrypted file E_d. The result is divided into n chunks and archived using a *Tape Archive Folder*. A final *xor* operation of the archive and a pseudo-random key concludes the *madEncrypt* scheme. *madEncrypt* can be compounded as follows:

$$E_d \ = \ \{ \ d \ \oplus \ K_H \ \} \ + \ R(n)$$
$$C_E \ = \ E_d \ / \ C$$
$$T \ = \ C_E \ \oplus \ K_{HC} \ ,$$
$$where \ K_H \ and \ K_{HC} \ is$$
$$H(k) \ = \ - \sum_{k \ \in \ \mathcal{K}} p(k) \ \log_b \ p(k),$$
$$where, \ b \ = \ 2$$

Let the initial encrypted data layer be, $E_d = (d \oplus K_H) + R(n)$, where d is the data to be encrypted within the initial encryption layer, K_H is the encryption key as a function of $H_i(k)$[9], and $R(n)$ as

[9]Shannon Entropy

a *Rubbish* block of size $1/n \times \text{len}(E_d)$. *Let* C_E be the deconstruction of E_d as $E_d \ / \ C$, where C is the chunk size in KB[10]. So that T is a Tape Archive[11] file containing the final encrypted file as $T = C_E \oplus K_{H_C}$, where K_{H_C} is the key as a function of $H_f(k)$.

In Figure 4, *Alice*'s Private Operations are enclosed with a *yellow* dashed box. *Alice xor*'s the file with a generated pseudo-random key. *Rubbish* of size $1/n$ $len(E_d)$ is added to the encrypted file E_d. The result is divided into n chunks and archived using a *Tape Archive Folder*. A final *xor* operation of the archive and a pseudo-random key concludes the *madEncrypt* scheme.

The decryption scheme(*madDecrypt*) is the supplemental transposition of *madEncrypt*, where the decryption of the T is followed by the decryption of the correct assembly of C_E given a set of instructions from a *Segment Identifier* parameter S_C.

madDecrypt can be compounded as follows:

$$\begin{aligned} C_E &= T \oplus K_{HC} \\ C_E \bowtie S_C &= E_d \\ d = \{E_d \oplus K_H\} &- R(n) \end{aligned}$$

A prevailing draw back of *xor* based encryption schemes is the key size. The *xor* cipher requires that both the data to be encrypted and the key to be of the same size. *madXOR* attempts to avoid the same size pairs by introducing *Rubbish* into the initial *xor* operation. *Rubbish* is pseudo-randomly generated data of an arbitrary length which is appended to the end of the initial *xor* operation in *madEncrypt*. The loop assumes a given size of $1/n$ the length of E_d where E_d is the encrypted file. In figure 5, the *randFunction* takes the file as *getFile* and loops over it given the file size as *getSize*$/n$ where n is a variable supplied by the instigator.

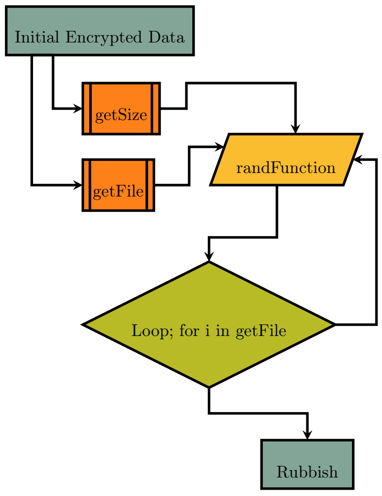

Figure 5: Rubbish Loop

3.3 Segment Identifier

Although inspired by Lamport's work in principle, the *Segment Identifier*'s role is not limited to the establishment of a digital signature whereby the preservation and integrity of the information object is upheld. Rather, various integral elements of **Tidy** are housed within the *Segment Identifier* to insure the validity of the scheme including the *madXOR* key as K_{H_C}.

The elements are as follows:

[10]KiloByte

[11]A Tape Archive folder: Known as a tarball with extension `.tar`, is a file storage format popular in Linux/Unix ecosystems

- *seg id*: A Segment Identifier is a pseudo random number of length 512[12] digits which will be used by the Formula Agreement scheme as a constant to encrypt the *Segment Identifier.*
- *master hash*: A cryptographic hash of size 512 bits of the *Segment Identifier*'s *sig* elements.
- *seg*: The Segment houses the operational elements within the *Segment Identifier.*
  - *key*: The *madXOR* Key as K_{H_C}.
  - *raw key*: The *madXOR* Key as K_H.
  - *chunk size*: The *madXOR* chunk size in KB as C.
  - *chunk order*: The *madXOR* chunk order in list form as S_C.
  - *next number*: The next number to abide by given the *Formula Agreement Sequence Condition.*
  - *num def*: The next Numerical Substitution Definition; more in §3.4.
  - *rubbish*: The rubbish block of size $1/n \times len(E_d)$.
- *sig*: The signature element.
  - *hata*: A cryptographic hash of size 512 bits of the *madXOR* encrypted file as T.
  - *ts*: The timestamp of the operation.
  - *linking*: Linking hash of the previous *seg id.*
  - *user salt*: Given an online agreed upon fingerprint[13], the *user salt* is a cryptographic hash of size 512 bits of *hata*, *ts*, and the $fingerprint$.

```
{
  "seg_id": Random_int_of_length_512,
  "master_hash" : sha512(Segment_Identifier)
  "seg": {
    "key": madXOR_key_as_KHC,
    "raw_key": madXOR_key_as_KH,
    "chunk_size": Chunk_Size_as_C,
    "chunk_order": Chunk_Order_as_Sc
    "next_number": Random_number_of_length_greater_than_48,
    "num_def" : Next_Numerical_Substitution_Definition,
    "rubbish" : Rubbish_Block,
    "sig": {
        "hata": sha512(Encrypted_Archive),
        "ts": timestamp,
        "linking": Linking_data,
        "user_salt": User_Salt
    }
  }
}

```

Listing 1: *Segment Identifier* Example Within The Private Operations Domain

The entirety of the *Segment Identifier* is encrypted with the Formula Agreement[14] scheme and is pushed to a public directory by *Alice* to be fetched by *Bob.* The contention of using a public directory to house the *madXOR* key and the next *Formula Agreement*'s sequence condition is alleviated by an absurd amount of valid[15] but mock *Segment Identifier*'s which are pushed with the original one as a means of complicating *Eve*'s offline attack. It should be noted that the example in Listing 1

[12]The length is somewhat arbitrary, during testing it was found that with small numbers, minor alterations will yield the same result given the Formula Agreement operation §3.4, hence the utilization of large numbers. It should be noted that the lengths have been chosen as to keep with the digital convention of base 2; as it would have been counter intuitive to represent the lengths otherwise.

[13]This could be any identifying label known by *Bob* that belongs to *Alice* and vice-versa.

[14]§3.4

[15]Valid in the sense that the Formula Agreement operation in §3.4 is mathematically correct.

is only visible within the Private Operations Domain; the publicly available *Segment Identifier* as shown in Listing 3 is the version which will be accessible to *Eve* and any other online party within the Public Operations Domain.

3.4 Formula Agreement

The Formula Agreement relies on various operations and constants. The main encryption scheme is a Modulo operation[16] whereby

$$n_i \times e \mod k = enc/n_i \mod k$$

where n_i is the subject of agreement prior to initiating communication, e is the *Segment Identifier* list in integer form representing the Unicode characters, k is the constant key shared by all parties including *Eve* and is defined as *seg id* in §3.3, and *enc* as the encrypted Unicode integers. The representation of the *Segment Identifier* relies on an additional obfuscation scheme based on numerical substitution(Listing 2[17]). The notion being that since *Eve* will launch an offline brute-force attack to solve for n_i, the exact digits representing n_i are obfuscated by means of an agreed upon definition. The agreement of the numerical substitution between *Alice* and *Bob* are shuffled–along with n_i–with every new information object transfer. An example can be found in Figure 6.

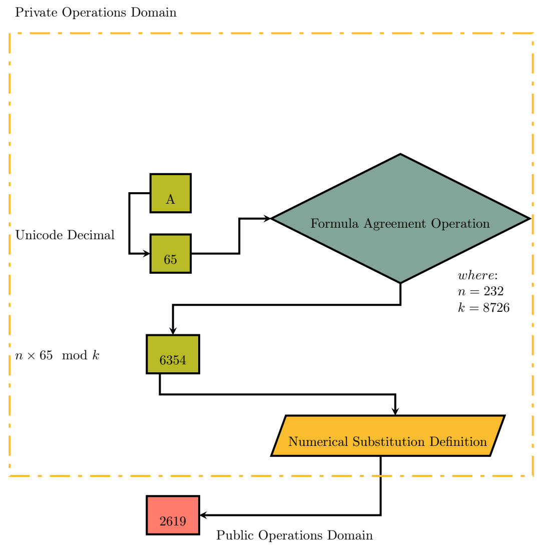

Figure 6: Abstracted Flowchart of Numerical Substitution Using The Example From Listing 2

```
{   "0": 0,
    "1": 7,
    "2": 3,
    "3": 6,
    "4": 9,
    "5": 1,
    "6": 2,
    "7": 4,
    "8": 5,
    "9": 8
}
```

Listing 2: Example of a Numerical Substitution Definition

[16]A discussion can be found in §4: *On The Matter of The Formula Agreement Operation*
[17]Note that 0 should not be substituted as to counter against frequency analysis attacks.

The Formula Agreement also includes an agreed upon formula to generate the *next number* as n_{i+1}; dubbed as the *Formula Agreement Sequence Condition.* The *next number* field will not contain the proceeding n_{i+1}, but rather a variable integer which will be used to perform a mathematical operation given the agreed upon formula to generate the *next number*. The rationale being that since the objective is to safeguard against a chaining attack, if any given *Segment Identifier* was brute-forced, further computation is required to break the next *Segment Identifier.* This can be compounded as follows:

$$n_{i+1} = f(n_i, \textit{next number})$$

where $f()$ is the agreed upon formula. As the compound statement shows, n_i must be used in tandem with the *next number* to generate n_{i+1}. The condition being contingent on the reuse of the current n_i. To summarise, *Alice* and *Bob* need to agree on three fundamentals before initiating communication:

- An integer of a length greater than 48 digits as n_i.
- A formula which will act as a sequence generator for n_{i+1}.
- A numerical substitution scheme which will be used to obfuscate k and *enc*.

If n_i and the formula are agreed upon; and subsequently the *Formula Agreement Sequence Condition* is met, the *Segment Identifier* is encrypted using the Formula Agreement Operation so that when the *Segment Identifier* is pushed to the public directory, the only revealing element is the *seg id* as shown in Listing 3. The *seg id*, which is defined herein as k is a shared key which will be used by *Bob* to decrypt the *Segment Identifier.* k will also be numerically substituted as a means of obfuscation.

```
{
  "seg_id": Random_int_of_length_512,
  "encrypted_seg": [enc]
}

```

Listing 3: *Segment Identifier* Example Within The Public Operations Domain

3.5 Tidy in Practice

Alice and *Bob* meet in person for the first time and agree on the following fundamentals:

1. An integer of length greater than 48 digits. as n_i.
2. A simple mathematical formula where n_i and n_{i+1} are variables.
3. A numerical substitution scheme such as the one described in Listing 2.

When *Alice* decides to contact *Bob* whilst under a presumed active session hijack, she initially starts by preparing the message to be encrypted in the form of a file. The file passes through the *madXOR* scheme and the resultant outputs are then carefully stored for further processing. The encrypted archive as T is sent to *Bob* after passing through the compromising intermediary *Eve.* The *madXOR* keys; K_{HC} and K_H, chunk information, rubbish, and the hash of the encrypted archive are all passed to the *Segment Identifier. Alice* starts constructing the *Segment Identifier* by adding the information objects from *madXOR*, a new numerical substitution definition, an arbitrary integer as *next number*, a generated integer greater than n_i as k, an operation timestamp, and a *user salt.*

After the construction of the *Segment Identifier* a final hash of the *Segment Identifier* is computed and appended to itself. *Alice* then uses the *Formula Agreement Operation* to encrypt the Unicode decimal representation of the of the *Segment Identifier* excluding the *seg id.* The resultant contains–as shown in Figure 3–two dictionary keys; the values of which will go through the

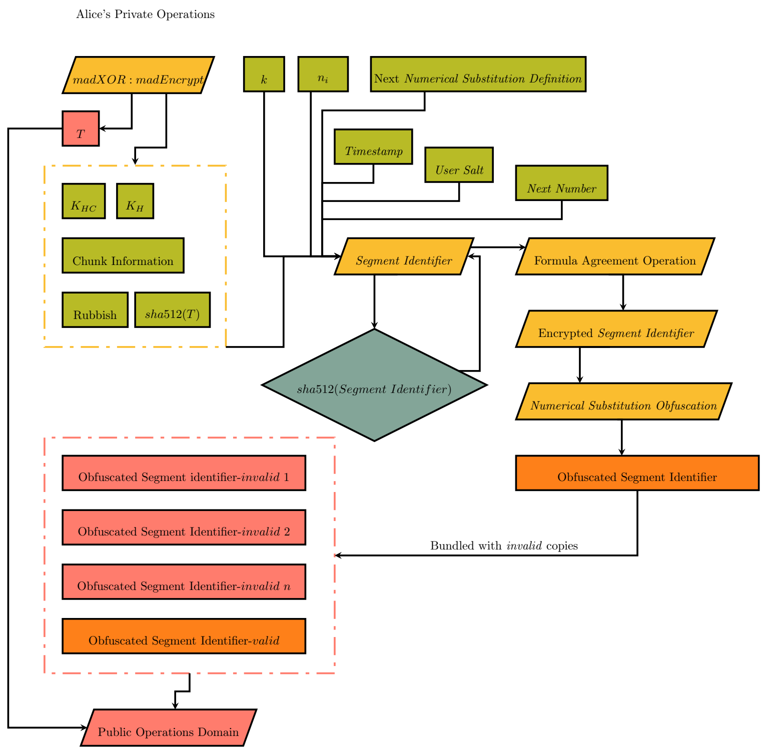
Alice's Private Operations
$madXOR$: $madEncrypt$
T
k
n_i
Next *Numerical Substitution Definition*
Timestamp
User Salt
Next Number
K_{HC}
K_H
Chunk Information
Rubbish
*sha*512(T)
Segment Identifier
Formula Agreement Operation
Encrypted *Segment Identifier*
*sha*512(*Segment Identifier*)
Numerical Substitution Obfuscation
Obfuscated Segment Identifier
Obfuscated Segment identifier-*invalid* 1
Obfuscated Segment Identifier-*invalid* 2
Obfuscated Segment Identifier-*invalid* n
Obfuscated Segment Identifier-*valid*
Bundled with *invalid* copies
Public Operations Domain

Figure 7: Tidy in Practice: *Alice*

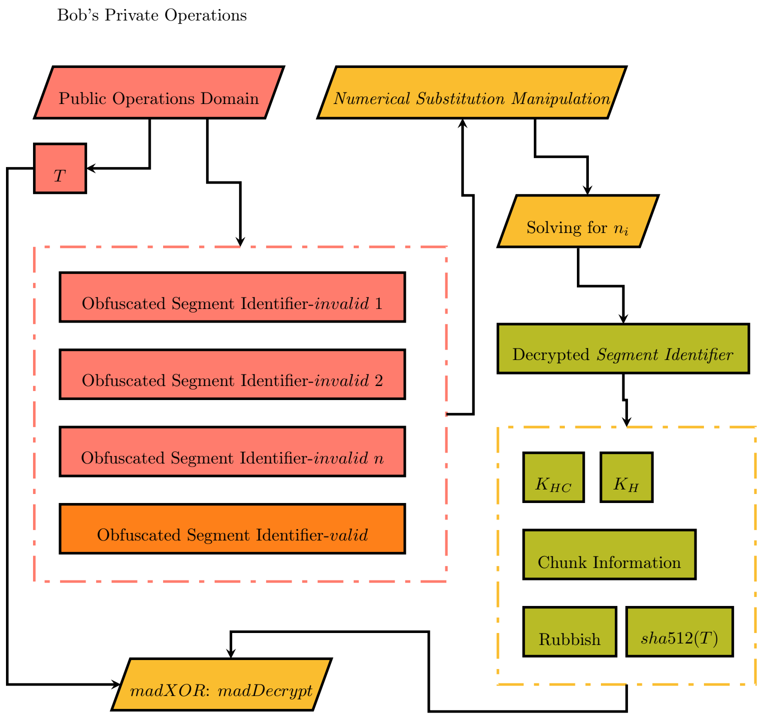
Bob's Private Operations
Public Operations Domain
Numerical Substitution Manipulation
T
Solving for n_i
Obfuscated Segment Identifier-*invalid* 1
Obfuscated Segment Identifier-*invalid* 2
Obfuscated Segment Identifier-*invalid* n
Obfuscated Segment Identifier-*valid*
Decrypted *Segment Identifier*
K_{HC}
K_H
Chunk Information
Rubbish
*sha*512(T)
$madXOR$: *madDecrypt*

Figure 8: Tidy in Practice: *Bob*

Numerical Substitution Definition obfuscating the encrypted numbers. *Alice* bundles the correct *Segment Identifier* with arbitrary copies of invalid *Segment Identifiers*', and pushes them to a public accessible directory. The latter can be shown in flowchart form in Figure 7.

Once *Bob* receives the encrypted archive as T, he accesses the public directory to retrieve the *Segment Identifier*s and starts solving for n_i after manipulating the the values of all the *Segment Identifier*s' using the *Numerical Substitution Definition.* After identifying the correct *Segment Identifier*, *Bob* retrieves the appropriate information objects and processes the encrypted archive as T to decrypt it. When *Bob* decides to respond, he goes through the same process as *Alice* with the exception of computing n_{i+1} using the agreed upon formula. The latter can be shown in flowchart form in Figure 8.

4 Remarks and Limitations

On The Matter of Public Key Encryption

It has been stated in *The Problem*[§2], that the use of a symmetric encryption scheme for the safeguarding of the file under the assumption of an active attack was not recommended. This is due to the assumption that neither *Alice* nor *Bob* were authenticated prior to *Eve*'s introduction. The use of a key exchange scheme such as the Diffie-Hellman key exchange might seem as the logical solution in such a case. However, within the main question is the assumption a fully compromising *Eve* and in turn a compromised key exchange. The use of an asymmetric public key exchange was also dropped for no implicit reason other than offering something new to the community.

On The Matter of The Public Segment Identifier

It might not seem obvious as to why the *Segment Identifier* is made public; if the encrypted file is sent directly to *Bob*, surely there is no reason why the *Segment Identifier* needs to be made public. The rationale for opting to publicly share it derives from a limitation within **Tidy**.

The limitation is size. For an initial file of 3 *kb*, the *Segment Identifier* could reach as high as 8 *mb* and as low as 4 *mb*. This is a problem. Although it might be considered a security measure given the order of magnitude in increased size and therefore a much more complicated offline attack by *Eve*, it is still a major hurdle and a nuisance for the recipient. Therefore it was decided that the best solution was to use an online service that might house large files rather than sending it directly to *Bob*. The service in question could be a cloud service, a torrent service, or any service that supports large uploads and is publicly accessible by anyone. *Alice* will not just send one *Segment Identifier*; as a normative arbitrary length, during testing no less than 10 copies including the correct one should be sent. Meaning that if we were to assume a raw file–the file to be encrypted–is of size 3 *kb*, *Alice* would be sending around 60 *mb* of *Segment Identifier*s.

On The Matter of The Formula Agreement Operation

As described in §3.4, the Formula Agreement Operation relies on a Module operation where

$$n_i \times e \mod k = enc/n_i \mod k$$

where n_i is an agreed upon integer, e is Unicode decimal, enc is the encrypted Unicode decimal, and k is a constant shared by all parties. The formula was chosen due to its simplicity. As a case in point example, *let* $n = 129873789$, e is the Unicode decimal representation of A as 65, and $k = 1233298789678677$. The latter will yield the integer 8441796285 as follows:

$$(129873789 \times 65) \mod 1233298789678677 = 8441796285$$

assuming that the *Numerical Substitution Definition* will not be applied, to decipher enc as 8441796285, the operation relies on a simple arithmetic operation:

$$(8441796285/129873789) \mod 1233298789678677 = 65$$

The limitation to the Formula Agreement Operation is that for it to function correctly, large integers for n_i and k are needed. This is principally due to the Modulo operation being the remainder of a division.

On The Matter of Obfuscation

Obfuscation embodies the transmission element of **Tidy** in the sense that several elements within the scheme are obfuscated and more may be obfuscated depending on the severity of the attack vector. However, this section is interested in the obfuscation elements of the *Segment Identifier* and subsequently the *Formula Agreement.* During the initial testing phase, the use of *AES*[18] was considered in combination with *RSA*[19] public key system; however both were dropped due to a prevailing drawback of variability. The initial approach to the **Tidy** question was based on a dynamic key. The idea was to circumvent against the unlikely possibility of exposing both *Alice* and *Bob* by breaking a singular message. Further testing was conducted and various solutions came in the form of dynamic Convolution, and the use of Keybase's[20] key directory as a means of accessing one's public keys. Both options were also dropped due to the computational overhead in the case of Convolution, and the dependency on a third party service in the case of Keybase. Moreover, the use of public key encryption as a whole was not a direct choice seeing as how it did not add anything new to the community. The working principle of the *Segment Identifier* is to solve for variability whilst keeping the scheme grounded in simplicity. However, the *Segment Identifier* alone does not offer any security to the scheme since it is basically a dictionary of keys and instructions to decrypt the intended file. The *Formula Agreement* scheme which acts as a security measure for the *Segment Identifier* was a needed mode of operation to safeguard against the presumed attack by *Eve.* To further impede the attack and in turn add a level of congruent conformity to the scheme, the *Segment Identifier* itself is obfuscated by bundling it with an array of invalid *Segment Identifier*s.

[18]Advanced Encryption Standard
[19]Rivest–Shamir–Adleman
[20]Keybase is an online public key directory.

References

[1] Leslie Lamport. Constructing digital signatures from a one way function. 1979.

[2] Björn Lundgren and Niklas Möller. Defining information security. *Science and engineering ethics*, 25(2):419–441, 2019.

[3] Scott J Warren and Greg Jones. Yokoi's theory of lateral innovation: Applications for learning game design. *Journal of Educational Technology*, 5(2):32–43, 2008.