Science and Global Security, 15:185–209, 2007

ISSN: 0892-9882 print / 1547-7800 online
DOI: 10.1080/08929880701715076

Routledge
Taylor & Francis Group

Protecting Nuclear Safeguards Monitoring Data from Tampering

Roger G. Johnston,[1] Michael J. Timmons,[2] and Jon S. Warner[1]

[1]Nuclear Engineering Division, Argonne National Laboratory, Argonne, IL, USA
[2]Current address: University of St. Andrews, St. Andrews, Fife, Scotland

Effective treaty monitoring requires that nuclear monitoring data be safe from tampering. Tamper-indicating seals and standard data encryption/integrity techniques, however, do not provide sufficient security, especially against electronic and physical attacks in the context of international nuclear safeguards. This article presents an alternative approach for assuring the integrity of monitoring data called the "One-Time Pad of Digits Substitutions" (OPODS). This cipher is a combination of the unbreakable one-time keypad, and the traditional substitution cipher. OPODS provides unbreakable security prior to an adversary trespassing inside nuclear monitoring hardware (even if the trespassing goes undetected), and good security after.

INTRODUCTION

International nuclear safeguards ("treaty monitoring") is a highly unconventional type of security application.[1] Unlike ordinary security applications such as *domestic* nuclear safeguards, the adversary in international safeguards (i.e., the "host" or nation-state being monitored) owns the assets and facilities of interest. This adversary will—for reasons of safety, security, counterintelligence, reciprocity, nationalism, and geopolitics—often insist on a full, detailed understanding of the monitoring strategies, hardware, and security techniques being employed by the inspectors. With conventional security applications, in contrast, adversaries are not ordinarily briefed. Other attributes of international

Received 5 March 2007; accepted 24 July 2007.
This work was performed under the auspices of the United States Department of Energy (DOE), LAUR-07-0884. The views expressed here are those of the authors and should not necessarily be ascribed to Los Alamos National Laboratory, Argonne National Laboratory, DOE, or the United States Government.
Address correspondence to Roger G. Johnston, Ph.D., CPP, Vulnerability Assessment Team, Nuclear Engineering Division, Argonne National Laboratory, 9700 South Cass Avenue, Building 206, Argonne, IL 60439-4825, USA. E-mail: rogerj@anl.gov

nuclear safeguards are also quite unique compared to domestic nuclear safeguards and other more routine kinds of security applications.[1]

Reliable treaty verification particularly requires that the inspectors can trust the veracity of the monitoring or surveillance data gathered in or near a nuclear facility by the monitoring hardware, which is typically unattended once running. Such hardware can include, for example, seismometers, radiological measuring instruments, video or photographic surveillance systems, access control devices, vehicle counters, and intrusion detectors. However, how can such data be secured, especially when the adversary in international nuclear safeguards has full access to technical details of the monitoring hardware, plus national- or world-class resources and technical expertise that can potentially be exploited for cheating?

Traditionally, tamper-indicating seals[2–4] are used to detect the opening of, or tampering with, hardware, electronics and instrumentation racks, while standard data encryption/integrity techniques are used to protect the recorded or transmitted data. Unfortunately, as will be discussed later, these approaches are not fully reliable for international nuclear safeguards.

This article presents a new technique (involving a combination of two previously unconnected old techniques) for guaranteeing data veracity called "OPODS" for "One-Time Pad of Digit Substitutions." OPODS is a simple, fast, and highly secure technique for securing recorded or transmitted data.[5] It is computationally practical for low-cost microprocessors. OPODS has no proprietary, licensing, or export control issues associated with its use (unlike many other modern encryption or authentication techniques), and it is well suited for use in international nuclear safeguards, in addition to certain other kinds of data logging applications where security is critical, yet where the adversary may understand details of the monitoring hardware, and be able to gain surreptitious physical access to it. An example would be a Global Positioning System (GPS) location data logger.[6]

WHAT'S WRONG WITH SEALS?

The Vulnerability Assessment Team (VAT) (formerly located at Los Alamos National Laboratory) has extensively studied hundreds of different tamper-indicating seals over the past 15 years. The VAT has shown how all the seals studied can be defeated quickly, using only low-tech tools, supplies, and methods that are available to almost anyone[2,7–9] (To "defeat" a seal means to remove it, then after stealing or tampering with the container contents, resealing using the original seal or a counterfeit, all without being detected.) The VAT has not yet seen a seal, including passive and electronic seals used for nuclear safeguards that require a sophisticated adversary or attack to defeat it.

There are practical countermeasures for most of the seal defeats demonstrated. These typically require the seal user to modify how he or she installs

and inspects seals. There also needs to be extensive hands-on training for seal installers and inspectors so that they understand the vulnerabilities for the specific seals they are using, and know how to look for the most likely attack scenarios. Unfortunately, all of this involves more time, money, and effort than most security programs are willing to invest, even for domestic or international nuclear safeguards.

Fortunately, better seals are possible.[2,10] Conventional seals can be modified to make attacks more difficult. An even more effective approach is to use a fundamentally novel approach to tamper detection, what the authors call "anti-evidence" seals.[2,10] Conventional seals can often detect tampering just fine, but must store this information in or on the seal until the seal can be inspected. However, an adversary can too easily hide or erase this "alarm condition," or replace the seal with a fresh counterfeit.

With anti-evidence seals, information is stored in or on the seal at the time it is installed that indicates that tampering has not yet occurred. If tampering is detected, this secret "anti-evidence"—typically a byte or two—gets instantly erased.[11] The absence of the anti-evidence at the time of seal inspection indicates that tampering has occurred. With this antievidence approach, an adversary cannot hide or erase the "alarm condition"; counterfeiting the seal hardware gains him nothing if he does not know the anti-evidence data to store in or on the seal.[12]

The authors believe anti-evidence seals can provide much more reliable tamper detection than conventional seals. In the authors' experience, however, there is relatively little serious interest in any quarter in better tamper-indicating seals, including for nuclear safeguards applications (domestic or international). Moreover, it is unlikely that anti-evidence seals, even if they prove superior to conventional seals, can provide absolute guarantees about detecting tampering. Seals should continue to be used, but they are not a panacea for assuring that monitoring data (or equipment) are free from tampering.[13]

WHAT'S WRONG WITH CONVENTIONAL TECHNIQUES FOR DATA INTEGRITY?

The critical issue of concern in this article is the integrity of the monitoring data gathered for verifying a nation's compliance with nuclear treaties, agreements, and international obligations.[14] Data integrity is commonly checked using hashes (encrypted or unencrypted) or closely related Message Authentication Codes (MACs).[15–17] Hashes are fixed length numbers, computed from a larger data set, that typically change dramatically when some or all of the original data set is modified. A checksum—summing all the digits—is an example of a simple hash.

Full encryption of the data (using a cipher) can also be used (although this is uncommon) to check data integrity in situations where the adversary

may want to make specific, unauthorized changes to the data in order to hide evidence of cheating.[18] Ciphers are more typically applied nowadays to data confidentiality applications where someone wishes to transmit data between two *physically secure* locations[19] such that anyone intercepting the encrypted data cannot understand what it means. The original secret data (or message) is commonly called the "plaintext," whereas the encrypted data (or message) is called the "ciphertext." For most applications involving the use of ciphers to provide data confidentiality, the bad guys are assumed to know the ciphertext and the encryption algorithm, but not the plaintext or the secret encryption key(s). This is not the case for international nuclear safeguards, where the adversary will typically know the plaintext for reasons discussed later and where data confidentiality is not desirable for reasons of transparency.

One of the reasons to consider using a cipher for data integrity (even though this is uncommon) is that it is generally more difficult to "break" a sophisticated cipher, that is figure out the secret key (even knowing the plaintext), than it is to create a new document that has the same hash or MAC tag as the original. Moreover, the level of effort required to break a cipher is usually well understood, whereas the security provided by a hash or MAC is often unclear.[20]

There are four main problems with using hashes, MACs, or conventional ciphers for guaranteeing the veracity of nuclear monitoring data. First, modern techniques that rely on secret keys (whether hashes, MACs, or ciphers) are only "computationally secure," not absolutely secure.[15–17,21] This means that experts *think* that figuring out the secret key or being able in other ways to create false data that appears authentic should require enormous mathematical and computational resources. The problem in international nuclear safeguards, of course, is that the nuclear adversary (being a nation-state) typically will have substantial resources, including world-class mathematicians, cryptoanalysts, and computers, or at least access to them. Moreover, historically hashes (encrypted or not), MACs, and ciphers that were once thought to be computationally secure have been defeated (sometimes surprisingly easily) as new cryptoanalytic techniques, computation power, and expertise become available.[22]

A second serious problem with hashes, MACs, or conventional ciphers is that they provide no significant security if the adversary can gain access to the sending or receiving location. This allows him to obtain the secret key(s) and/or algorithms being used, or to directly tamper with the plaintext, and thereby replace the real data with his own fake data. In theory, if trespassing into monitoring hardware could be reliably detected and the encryption key(s) or security algorithms erased quickly enough, the adversary would have a more difficult challenge in faking the monitoring data (although it would still be theoretically possible). Seals and current tamper detection methods, however, are not up to the challenge, as previously discussed. In addition, the time to thoroughly erase modern cipher keys can be relatively long on microprocessors

because the minimum recommended key size for high security applications is 2048 bits = 256 bytes.[23,24] MAC keys tend to be smaller, but still require a minimum of 128 bytes to be quickly and reliably erased, and often more for high level security.

A third problem with conventional methods is that even if the secret key(s) get fully erased before an adversary can retrieve them, the adversary in international nuclear safeguards will typically know the plaintext, the ciphertext, and the cipher or data authentication algorithm being used.[25] He or she will know the plaintext because it is his facility where the monitoring measurements are made and he will understand what is occurring in his own facility. He will also be able to get the ciphertext, that is, the stored logged data, and (if necessary) the data encryption/integrity algorithm by trespassing into the electronics or otherwise hacking the microprocessor. In fact, he may automatically know the algorithm because the inspectors will probably be required to disclose it for reasons of transparency, and because the inspected nation may insist on being provided with the software source code.[26] In this case a cipher is much easier to break (i.e., figure out the secret key) when the adversary knows the plaintext, the ciphertext, and the cipher algorithm[27]—a situation that is atypical for conventional encryption applications (although common for hash or MAC applications).

The fourth problem with hashes, MACs, and conventional ciphers is that they are computationally intensive and can be difficult to implement efficiently on a microprocessor,[28] making them less than practical for small, cost-effective, transparent field monitoring equipment.[29]

THE ONE-TIME KEY PAD

There is only one cipher that can be proven mathematically to be unbreakable for all time: the One-Time Key Pad.[21,30] This cipher, also known as the one-time pad (OTP) or Vernam cipher, was invented around 1917. In addition to being unbreakable, the OTP has the advantages that it has no proprietary, licensing, or export control issues (unlike some modern ciphers and MACs), and is simple and very fast.

The idea with the OTP is to use a totally random key that is as long as the plaintext. This key can be used only once, and then must be discarded.[31] Although Soviet spies used the one-time keypad extensively in the 20th century,[32] it has not been considered practical for many applications because of the large storage requirements for the key. With the ever decreasing cost and size of digital storage media, however, this disadvantage is becoming much less important. The OTP is typically used to encrypt alphabetic characters, but here its use is demonstrated for encrypting numeric values, which are more relevant for monitoring data. Assume one wishes to encrypt the digits “1663,” and

that the first 4 random digits in the one-time pad or random digits are "3907". Thus:

plaintext	1	6	6	3
OTP	3	9	0	7

Adding the plaintext and OTP numeric values, the results are

plaintext + OTP sum	4	15	6	10

Sum values in excess of 9 are "wrapped around" by subtracting 10, that is, the modulus 10 is computed[33] to obtain,

sum mod 10:	4	5	6	0

The encrypted message (ciphertext) is thus "4560." To decrypt the message, the process is reversed by subtracting the OTP values using modulus 10.

Note that the OTP cipher alone is not sufficient for securing safeguards monitoring data for two reasons. First, it is difficult for a microprocessor to rapidly erase a large one-time pad (or the ciphertext) once physical or electronic intrusion is detected. Secondly, even if the one-time pad gets erased, the adversary can trivially reconstruct it because he knows the plaintext and can read the stored ciphertext. Reconstructing the OTP allows him to replace the true monitoring data with fake data that will appear authentic to the nuclear inspectors.[34]

SUBSTITUTION CIPHER

Even simpler than the one-time pad is the substitution cipher,[17,35] which is thousands of years old, and thus has no proprietary, licensing, or export control issues. This cipher, however, can be easily broken, even by amateurs. As the name suggests, a substitution cipher involves substituting each plaintext character with another predetermined ciphertext character. For example, assume that the substitution cipher is given by the following one-to-one correspondence between each plaintext decimal digit in the top row with the digit directly below it in the bottom row:

plaintext digit	0	1	2	3	4	5	6	7	8	9
mapping	2	9	8	3	7	1	0	6	5	4

The (secret) bottom list of digits in random order (with none repeating) specifies the substitution cipher, and is known as the key or "mapping" because it specifies how the plaintext digits "map" to ciphertext digits. For example, in the above cipher, 1 maps to 9, 6 maps to 0, 3 maps to 3, and so on. Thus,

plaintext "1663" (for example) encrypts to ciphertext "9003." To decrypt, the mapping (key) is used in reverse.

In a traditional substitution cipher, the mapping is unchanged, no matter how long the message. A given plaintext digit will always encrypt to the same ciphertext digit, regardless of where the digit appears in the plaintext.[36] This is what makes the security of a substitution cipher so poor, in contrast to the OTP where the (additive) key is always changing.[37] Like the OTP, the conventional substitution cipher is useless on its own for protecting monitoring data—especially when the adversary can break into the monitoring hardware without being reliably detected.

ONE-TIME PAD OF DIGIT SUBSTITUTIONS (OPODS)

OPODS is an encryption method that is the union of a one-time pad and a substitution cipher. It can be thought of as a substitution cipher where the random mappings (or "key") change for every character or digit to be encrypted. The encryption step is done in the same way as the substitution cipher; the only difference is the mapping changes after every plaintext digit is encrypted, and each mapping is immediately erased after it is used. Another way to think of OPODS is as a one-time pad of substitution mappings, each mapping being discarded after it is used just once to encrypt one character or digit.[38]

Consider the following example where the (plaintext) data "1663" is encrypted. The first mapping shown below is used to substitute "2" for the first digit ("1") in the plaintext; "2" thus becomes the first ciphertext digit. To find the substitution for the second digit, one moves to the next random mapping after erasing the first. This means that "6" in the plaintext gets replaced by "8." This process is continued until the plaintext is all encrypted.

Mapping for first plaintext digit:

plaintext digit	0	1	2	3	4	5	6	7	8	9
mapping	9	2	4	1	7	3	6	5	0	8

So, for example, "**1**" → "**2**"

Mapping for second plaintext digit:

plaintext digit	0	1	2	3	4	5	6	7	8	9
mapping	2	4	0	1	5	7	8	6	9	3

So, for example, "**6**" → "**8**"

Mapping for third plaintext digit:

plaintext digit	0	1	2	3	4	5	6	7	8	9
mapping	9	2	1	7	5	3	0	6	8	4

So, for example, "**6**" → "**0**"

Mapping for fourth plaintext digit:

plaintext digit	0	1	2	3	4	5	6	7	8	9
mapping	1	2	6	7	0	8	5	9	3	4

So, for example, "**3**" → "**7**"
These mappings thereby give us the ciphertext "2807."

This OPODS method is just as strong (unbreakable) as the conventional one-time pad (OTP) for ciphertext only attacks. However, unlike the conventional OTP, if the OPODS ciphertext and plaintext are both known, it is still not possible for an adversary to reconstruct used, erased mappings even if unlimited amounts of plaintext and ciphertext are available. This is because the only information that can be confidently inferred is the value of just one of the 10 digits in each previously used mapping. Because the adversary wants to encrypt a different set of plaintext digits than were actually encrypted, he is stuck.[39] With no clues as to what the used mappings were, he has only a 1 in 9 chance[40] of guessing the correct ciphertext digit corresponding to each plaintext digit he wants to fake.

Now nuclear safeguards monitoring data will typically involve quantitative sensor readings recorded by a microprocessor. It thus makes more sense to use hexadecimal ("hex" or base 16) digits, rather than base 10 decimal digits for the plaintext, ciphertext, and OPODS mappings.[41] With the use of the hex digits, each OPODS mapping consists of 16 hexes, in random order, with no hex digit repeated in a given mapping.

Each of the OPODS hex mappings will therefore be a random permutation of the set of hex digits 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, F. There are over 20 trillion (16!) possible distinct mappings. By coincidence, one mapping can be identical to another, but only if it is unpredictably replicated.

OPODS DATA SECURITY

Now with OPODS, even if an adversary's trespassing goes undetected, he will not be able to fake data prior to his break-in (assuming the mappings are truly erased irreversibly as they are used). There is no need for any erasure of data in order to protect the veracity of previously recorded monitoring data.[42]

But what about faking future data? Unless all of the *unused* mappings can be erased instantly when trespassing is detected—and as discussed earlier, this cannot be guaranteed—then the OPODS mappings will be available to the adversary to fake future data.

The way to deal with this problem is to have the microprocessor pick each mapping as it is needed from a cache of unused mappings in a manner known only to the inspectors. This can most easily be done using a pseudo-random number generator (PRNG), a simple iterative equation that deterministically

and iteratively generates a new pseudo-random number from the previously generated number. The new pseudo-random number points to which mapping should be used next. The PRNG is initialized with a secret key ("seed") known only to the inspectors, although the PRNG algorithm itself need not be secret. One common form for a PRNG is a linear congruent generator which produces the pseudo-random sequence of integers $\{I_0, I_1, I_2, \ldots\}$ in the range [0, M-1] via the formula $I_{n+1} = (A\ I_n + C) \bmod M$, where the integer coefficients A, C, and M must be carefully chosen, and $I_0$is the seed.[43–45]

Figure 1 shows how this picking process might most effectively be implemented. The PRNG picks a mapping from a cache of (for example) 100 OPODS mappings. As each mapping is used, it is replaced by the next available mapping from a larger cache of mappings. The PRNG generates a 2-byte number, 0–65535. The modulus 100 of this value points to which of the 100 mappings

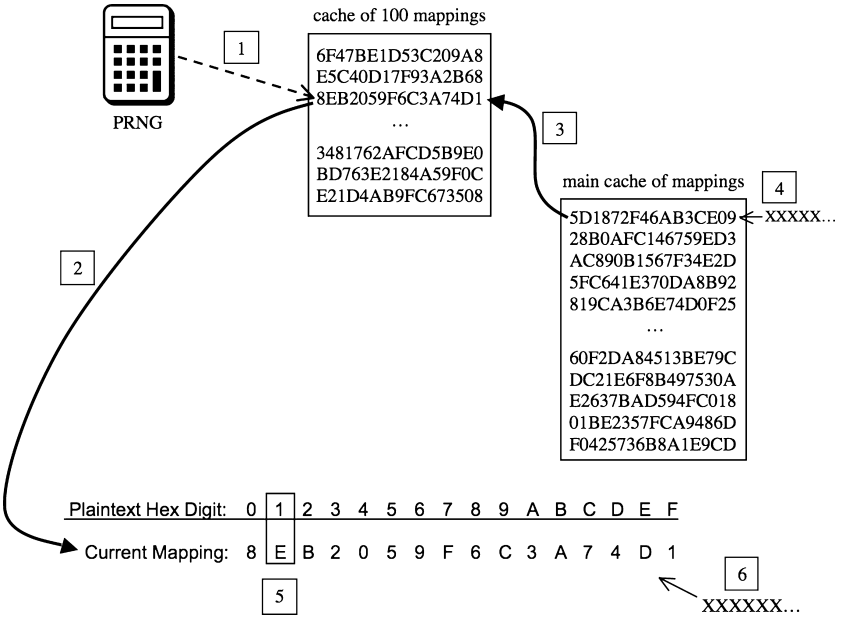

Figure 1: A schematic of the OPODS algorithm. In step 1, a pseudo-random number generator (PRNG) picks one of the random mappings from a cache containing 100 random mappings. In step 2, this mapping is moved to the location where it will be used as the current mapping for OPODS encryption of one hex digit. The moved mapping is immediately replaced in the cache of 100 by the next available mapping in the main cache (step 3). This mapping, in turn, gets immediately and irreversibly erased from the main cache (step 4). In step 5, the current mapping is used to change one hex digit of the plaintext (1 in this example) into one hex digit of ciphertext (E). Finally, the current mapping is erased (step 6). By the end of step 6, useful information about the current mapping is permanently lost to an adversary. If trespassing is detected, the current iterative value of the PRNG (typically 2 bytes in length) is immediately erased, leaving an adversary with no clues as to which of the 100 mappings in the cache of 100 was intended to be used next. For even better odds (for the good guys), more than 100 mappings can be stored in the smaller cache.

in the smaller cache to use next. If trespassing is detected, erasing the current 2-byte value of the PRNG leaves the trespasser with no clue as to the order in which future mappings would have been chosen, even though he will probably know the PRNG algorithm and maybe the current mapping.[46]

The security of post-trespassing monitoring data, however, is not as high as pretrespassing data, because for the former, one must detect the trespassing, plus know for sure that the PRNG's current 2-byte value has been irretrievably erased. Fortunately, however, such an erasure can typically be accomplished in 2 microseconds or less in a modest microprocessor.[47,48]

Note that erasing just 2-bytes with OPODS is much faster than erasing the 128-byte or 256-byte (or longer) keys for a MAC or a conventional high-security cipher, respectively.[49] This may have important security implications.[50] Moreover, with a standard keyed hash, MAC, or cipher, erasing the key does not guarantee data integrity because the hash, MAC, or cipher can theoretically be broken—especially in international nuclear safeguards where the adversary most likely knows the plaintext, the ciphertext, and the algorithm being used, and has enormous resources. Full erasure of the current 2-byte PRNG value, in contrast, leaves the adversary with no hope of reliably faking future data, no matter how sophisticated his cryptoanalytic capabilities.

OPODS STORAGE REQUIREMENTS

With 16 non-repeating hex digits in each OPODS mapping, there are $16! = 2.1 \times 10^{13}$ possible mappings. This corresponds to a minimum of 44.3 bits needed to represent any possible mapping.

As a practical matter, however, the simplest and fastest approach is to store each mapping directly as 16 hexes = 64 bits. The advantage is that the mapping is just a lookup table that requires no computation or decompression of the mapping by the microprocessor in the field.[51]

A slightly more efficient way to store the mappings, but that requires only modest extra computation by the microprocessor, is to store each mapping as 15 hexes instead of 16. The final hex does not need to be specified because it is the only hex digit not yet appearing in the mapping. This approach requires only 15 hexes = 60 bits per mapping.

Many other algorithms are possible for representing mappings that require less storage, but more computation. For example, after the first 8 hex digits in a mapping are specified, only 3 bits are needed to specify the next hex digit from the list of the $2^3 = 8$ remaining hex digits not yet chosen. After 12 hex digits are chosen, only 2 bits are needed, and after 14 are chosen, only 1 bit is needed. With this approach, 49 bytes are needed to fully specify a mapping. The disadvantage to this "Remaining Digits Algorithm" is that more microprocessor computation time is required to decompress each mapping as it is needed.

Table 1: Storage and decompression requirements for various algorithms for storing OPODS mappings.

Algorithm complexity	Bits per mapping	Bytes of storage needed per byte of plaintext	Decompression complexity
List 16 Hex Digits	64	16	None
List 15 Hex Digits	60	15	Minimal
Remaining digits	49	12.2	Moderate
High-Low	46.4[1]	11.6[1]	High
Theoretical minimum storage	44.3	11.1	Very high

[1]On average.

Another potential algorithm, called the "High-Low Algorithm," specifies each hex digit in the mapping by up to four bits. These bits indicate whether the hex digit in question is in the lower half of the ordered list of unchosen hex digits, or the upper half. The half that the hex digit does not belong to is then removed, and the question is asked again of the remaining ordered list of unchosen hexes. This is repeated up to four times until the hex digit is fully specified. As the mapping gets longer, the ordered list of unchosen hex digits gets shorter, and fewer than 4 bits are required to specify each hex digit. This algorithm requires, on average, 46.4 bits per mapping, illustrated in Table 1.[52]

Even with the least efficient storage algorithm (the "List 16 Hex Digits" Algorithm), OPODS does not require an impractical amount of storage. Each GigaByte (GB) of plaintext would need only 16 GB of OPODS mappings, and that storage gets freed up for other uses as each mapping gets used. Currently, the *retail* cost of 16 GB of storage is under $3 for magnetic hard disk storage and under $140 for flash memory. Not only are these prices lower for storage devices purchased in volume, but the cost of storage continues to drop precipitously over time, as shown in Figure 2.[53]

Another storage issue for microprocessors is the amount of random access memory (RAM) available. Most conventional ciphers that one might try to implement on a microprocessor require 50 to 2300 or more bytes of RAM, yet standard low-cost 8-bit microprocessors typically have either 128 or 256 bytes.[28] The OPODS scheme shown in Figure 1 would actually require only 2 bytes of RAM for the current PRNG iterative value.

A third storage issue is the amount of programming space required. Many modern encryption schemes do not fit on current commercial microprocessors[28] and advanced MAC algorithms are relatively large. With OPODS, in contrast, the encryption algorithm is just a lookup table. It requires only a few dozen BASIC programming lines, and only a few hundred machine instructions—the exact amount depending on the OPODS mapping decompression scheme chosen and the complexity of the PRNG being used.

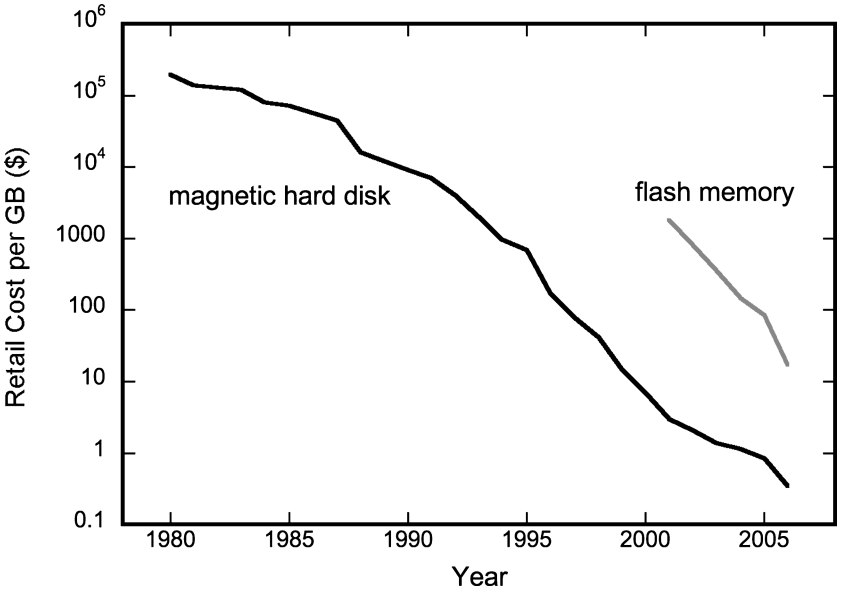

Figure 2: The decreasing retail cost over time of 1 GB of storage for magnetic hard disks and flash memory. If this trend continues—even without new storage technologies becoming available—it will be possible to store 1 GB of data nearly free by 2015.

OPODS SPEED

OPODS encryption was implemented on a Microchip PIC16F819 microprocessor ($4.00 quantity of 1) along with two 24FC515 (512K bit) memory chips ($4.68 each in quantity of 1). The software was written using the PICBASIC PRO compiler.[54]

With this system, the computation time required to encrypt 1 byte of plaintext is about 160 clock cycles, independent of the oscillator speed: 4, 8, or 20 MHz. (Thus it takes 8 microseconds per byte using a 20 MHz oscillator.) A more efficient algorithm, and the use of assembly language programming instead of the BASIC compiler, would undoubtedly increase encryption speed. By way of comparison, the highly optimized Twofish encryption algorithm requires 1820 clock cycles per byte on an 8-bit microprocessor, but offers relatively low levels of security and at most a 32-byte key.[55]

GENERATING OPODS MAPPINGS

Random OPODS mappings can best be generated from a one time pad (OTP) of random hex digits, which have themselves been generated nondeterministically using hardware (discussed in the next section). Generating the OTP from a deterministic PRNG method (such as found on most computers) does not result in true random digits and is not recommended because an

adversary may be able to predict its behavior, especially if large numbers of OPODS mappings are stored in the monitoring hardware.[56] PRNG security is weak because computer PRNGs are well studied, plus there are only a small number of other widely used PRNGs.[57] Indeed, there are known instances of people predicting PRNG values, thus "breaking" the PRNG.[58]

The problem with using an OTP of random hex digits to generate OPODS mappings is that each OTP digit is equally likely, yet every OPODS mapping must be made up of 16 random hex digits with each digit appearing exactly once. One possible algorithm for building an OPODS mapping from a one time pad involves simply appending the next random hex digit in the pad to the mapping if that hex digit does not yet appear in the mapping, and discarding the hex digit and skipping to the next random OTP hex digit in the pad if it does. This method is straightforward and produces very random OPODS mappings if the original OTP hex digits are random, but wastes an average of $60.644\% \pm 0.023\%$ of the OTP in order to avoid duplicate digits in a mapping. The algorithm is referred to as the "Waste Algorithm." An example is shown in Figure 3.

Another method for generating OPODS mappings from an OTP of random hex digits is called the "Not Yet Used Algorithm." See Table 2 for an example.

As the OPODS mapping is built, the authors keep track of which hex digits do not yet appear in the mapping. When a new mapping is started this "unused list" is pseudo-randomly scrambled using the computer's PRNG. Each hex digit in the OTP is then used to select which hex digit in the "unused list" will be appended to the mapping.[59] Once the mapping is complete with 16 non-repeating hex digits, it is started again on the next mapping with a freshly scrambled "unused list." This algrithm wastes none of the OTP, but is somewhat less secure in that it still relies on the computer's PRNG.[60]

Random OPODS mappings can also be created by letting the OTP values shuffle the hex digits {0123456789ABCDEF} in a random fashion using, for example, the Fisher-Yates Shuffling Algorithm.[61]

Another approach is called the "Reseeding Algorithm." It uses the hardware-generated OTP random hex values to intermittently re-seed a computer's internal PRNG or other PRNG. The values generated by the PRNG are

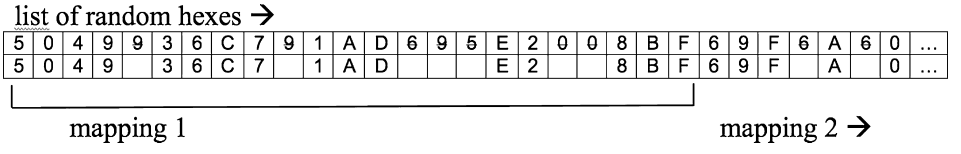

Figure 3: An example of the "Waste Algorithm" for generating random OPODS mappings (bottom row) from a hardware-generated one time pad of random hex digits (top row). Any time one of the hex digits in the OTP already appears in the mapping being built, it is skipped. In this example, the authors use the first 9 for mapping 1, but ignore the digit 9 the second, third, and fourth time it appears (which is why the duplicate 9s are shown crossed out). Once a mapping has its full complement of 16 non-replicated hex digits, the authors building the next mapping. In this example, the completed mapping 1 = {504936C71ADE28BF}, whereas mapping 2 begins {69FA0 . . .} .

Table 2: An example of the "Not Yet Used" algorithm for generating a random OPODS mapping from a one time pad of random hex digits.

Step	Random hex from OTP	Scrambled "unused list"	Mapping under construction
1	C	EB46C1572A0D39F8	C
2	1	EB461572A0D39F8	CB
3	0	E461572A0D39F8	CBE
4	F	461572A0D39F8	CBE1
5	0	46572A0D39F8	CBE14
6	A	6572A0D39F8	CBE148
7	4	6572A0D39F	CBE148A
8	2	65720D39F	CBE148A7
9	B	6520D39F	CBE148A70
10	1	652D39F	CBE148A705
11	6	62D39F	CBE148A7056
12	0	2D39F	CBE148A70562
13	7	D39F	CBE148A70562F
14	3	D39	CBE148A70562FD
15	5	39	CBE148A70562FD9
16	—	3	CBE148A70562FD93

In step 1, the authors just make the first hex digit in the new mapping equal to the next available random digit in the OTP (C in this case). In step 2, the next random hex digit from the OTP is 1. This is used to point to the hex digit in position 1 in the (pseudo-randomly scrambled) "unused list," which is B in this example, so B gets appended to the mapping. (The first digit in the "unused list" is defined as position 0.) In step 3, the hex digit in the "unused list" at position 0 is E. In step 4, the F position is position 15, but there are only 13 hex digits in the "unused list," so the authors wrap around to position 2, that is, the hex digit 1. This process continues until the 16th step, where the sole remaining hex digit in the "unused" list (3 in this example) is automatically appended to the mapping. To build the next mapping, the authors create a new pseudo-randomly scrambled "unused list" and new random hex digits from the OTP.

then used to build each random mapping. This algorithm is fast and makes highly efficient use of the OTP. It is not, however, maximally random because its behavior is dominated by a PRNG. Other algorithms for generating OPODS mappings apart from the ones considered here are also possible.

Regardless of the algorithm, the computation time for generating OPODS mappings can be relatively modest on a desktop computer.[62] Using an Apple 2GHZ Intel Core Duo Mac computer, with programs written in REALbasic,[63] the time to generate 2 million OPODS mappings from an existing OTP—enough to encrypt 1 Mega-Byte of plaintext—is 3.7, 1.1, and 3.1 minutes, respectively, for the "Waste Algorithm", "Not Yet Used Algorithm," and Fisher-Yates Shuffling (16 shuffles per mapping). A faster computer, plus more optimized algorithms and computer code would substantially decrease these times.

GENERATING TRUE RANDOM DIGITS

The unpredictability of OPODS mappings depends critically on the unpredictability of the OTP random digits from which they are built. Quantum effects probably produce the greatest randomness.[64] Fortunately, truly (or at least highly) random sequences can be generated a number of different ways

Table 3: Attributes of various means for physically generating random numbers in hardware. All of these, except for the lava lamp and some electronic noise techniques, are wholly or partially quantum mechanical in nature.[3]

Technique	Typical generation rate (bytes/min)	Typical cost[2]
Keyboard & mouse	2–100	$0
Lava lamp	10^1–10^6	$100–$600
Background radioactive decay	5–10	$300
Source radioactive decay	20–6,000	$350–$800
Plasma disk/sphere[1]	200–1000	$100
Radio noise	10^3–10^4	$500
Photons & beamsplitter	10^6–10^8	$2000
Electronic noise	2×10^4–2×10^8	$300–$3500

[1]For the values in this row, 4 independent photodiodes are used to monitor the light output from one plasma discharge disk or lamp.

[2]Typical cost for hardware and sensors in retail quantities, but not including the computer or interface hardware used to acquire and record the random numbers.

[3]Plasma discharge is a somewhat quantum phenomenon intrinsically, but the exact spatial and temporal behavior of the plasma discharge is probably additionally affected by cosmic rays, which are very quantum mechanical. Whether the keyboard & mouse technique is wholly or partially quantum mechanical depends on the extent to which human beings and their decisions are governed by quantum mechanical events.

with relatively low cost hardware. They include measuring: radioactive decay from either background radiation or a radioactive source[65]; electronic noise (thermal or Johnson noise, avalanche noise, other amplifier noise, and electron quantum tunneling)[66]; photons passing through a 50-50 beamsplitter[67]; noise in a shuttered video camera; microphone noise[68]; the light from a lava lamp[69]; radio-frequency noise (from de-tuned radios)[69]; computer hard disk noise[70]; ping-pong balls such as in machines used by lotteries[71]; and the unpredictability of the timing in how a human uses a computer keyboard or mouse.[72] Table 3 summarizes some of these methods.

Five of the eight methods listed in Table 3 are used by the authors for generating random numbers, but the easiest is the use of photodetectors to monitor plasma discharge disks and spheres that are sold as consumer novelty lamps (see Figure 4). Up to four different photodetectors can be used to generate separate, uncorrelated random sequences per lamp. This novel technique is safe, simple, and inexpensive, and can generate random numbers at moderately high rates. It is also quite immune from calibration and threshold drift problems, and from tampering at a distance. Sequences of digits generated in this way have consistently passed all the tests for randomness applied to them.

INSTALLING THE OPODS MAPPINGS

The OPODS mappings must be installed or delivered to the monitoring hardware by the inspectors in a manner that does not compromise its security. One method is to have the random mappings stored on a flash memory thumb drive

Figure 4: A consumer 2-dimensional Lumina Disk, left, and a 3-dimensional Buddha Spherical Lamp, right, of questionable taste. These are sold to consumers as retail novelties and decoration for approximately $19 and $25, respectively. The plasma discharge "fingers" that each device generates fluctuate unpredictably in time and space. Up to 4 different photosensors, if sufficiently spaced, can measure the light levels at different points on each device without any significant correlation or anti-correlation in their measurements.

that is copied into the monitoring hardware in the field by the inspectors in person when they first start it up. The thumb drive must then be fully erased, or else kept in the possession of the inspectors. If the host (inspected) nation is concerned about what information may be being passed into the hardware, a "choose or keep" scheme[73] can be employed wherein the inspectors lay out three or five thumb drives (each with different OPODS random mappings) and the host nation randomly picks the one to be installed, and gets to keep another one to reverse engineer. The latter will not be used for monitoring, but can instead be checked by the host to be sure it contains only random mappings, not microprocessor code or non-random data.

An inexpensive 4 GB flash memory thumb drive can store up to 364 million OPODS mappings, enough to encrypt 182 MB of plaintext data.[74] If one 2-byte (0-65535) measurement is made and recorded each second, this is enough OPODS mappings for almost 3 years of monitoring!

CONCLUSION

Improvements are needed to secure nuclear monitoring data given the current unreliability of tamper-indicating seals, and the less than guaranteed security offered by conventional data authentication methods, whether hashes, MACs, or ciphers. OPODS is an attractive alternative. Unlike conventional data integrity methods, OPODS-encrypted monitoring data that are recorded (or transmitted) prior to trespassing are fully secure—not just "computationally secure"—even if the monitoring hardware fails to detect the trespassing and nothing gets erased. Post-trespassing data is safe if the trespassing can be detected and a mere 2-bytes quickly erased. Other data integrity techniques, in contrast, require erasing at least 128 bytes, and sometimes considerably more.

OPODS has other advantages as well. It is very fast and computationally simple—basically an erasable lookup table. Being so straightforward and low-tech, OPODS is conducive to the transparency, simplicity, and high levels of host comfort that are so important for international nuclear safeguards. Its simplicity should also make OPODS relatively immune from adversarial "side channel" attacks such as timing or power analysis methods[75] and from chosen-plaintext attacks.[17] OPODS does not tie up large amounts of RAM, code space, or microprocessor computation time, and is quite practical for implementing on low-cost 8-bit microprocessors. Unlike some modern MACs and ciphers, OPODS has no proprietary, export control, or licensing issues. Moreover, its security is not compromised (unlike some other encryption or data integrity methods) if the same plaintext message is encrypted more than once.

The relatively large amount of data storage required for OPODS actually offers a security advantage. If an adversary were to use sophisticated techniques to try to reconstruct erased OPODS mappings—a significant concern[49]—so that he could change post-trespassing monitoring data, he would have a very large amount of information to try to recover, making the task more difficult.

The disadvantages of OPODS include the requirements to have 11–16 bytes of mapping data per byte of plaintext to be encrypted (although this storage space is freed up for other uses as ciphertext is generated). OPODS also requires large amounts of non-deterministic, hardware-generated random numbers, and installation of the OPODS mapping data into the monitoring hardware in a manner that keeps them secret. Unlike public-private key ciphers, OPODS is not conducive to 3rd party authentication.[76]

OPODS was implemented on both a desktop computer and a low-cost 8-bit microprocessor, and the authors found it easy to develop and use.

NOTES AND REFERENCES

1. See, for example, Roger G. Johnston and Morten Bremer Maerli, "International vs. Domestic Nuclear Safeguards: The Need for Clarity in the Debate Over Effectiveness," *Disarmament Diplomacy* 69 (2003): 1–6, http://www.acronym.org.uk/dd/dd69/69op01.htm; Morten Bremer Maerli and Roger G. Johnston, "Safeguarding This and Verifying That: Fuzzy Concepts, Confusing Terminology, and Their Detrimental Effects on Nuclear Husbandry," *Nonproliferation Review* 9 (2002): 54–82, cns.miis.edu/pubs/npr/vol09/91/91maerli.pdf

2. Roger G. Johnston, "Tamper-Indicating Seals," *American Scientist* 94 (2006): 515–523.

3. Roger G. Johnston, "Tamper-Indicating Seals for Nuclear Disarmament and Hazardous Waste Management," *Science and Global Security* 9 (2001): 93–112.

4. Daniel Engber, "How Do You Seal a Nuclear Plant?" http://www.slate.com/id/2123769/

5. Two important factors when considering the security of monitoring data that are transmitted in real-time include the lag time in receiving the data at headquarters (which can potentially be exploited by the adversary), and the fact that real-time data is often recorded and stored (at least temporarily) before being analyzed.

6. Jon S. Warner and Roger G. Johnston, "GPS Spoofing Countermeasures," Homeland Security Journal, December 12, 2003, http://www.homelandsecurity.org/ bulletin/Dual%20Benefit/warner_gps_spoofing.html

7. Roger G. Johnston, "Tamper Detection for Safeguards and Treaty Monitoring: Fantasies, Realities, and Potentials," *Nonproliferation Review* 8 (2001): 102–115, http://www.princeton.edu/~globsec/publications/pdf/9_2johnston.pdf

8. Roger G. Johnston and Anthony R. E. Garcia, "An Annotated Taxonomy of Tag and Seal Vulnerabilities," *Journal of Nuclear Materials Management* 229 (2000): 23–30.

9. Roger G. Johnston, Anthony R. E. Garcia, and Adam N. Pacheco, "Efficacy of Tamper-Indicating Devices," *Journal of Homeland Security*, April 16, 2002, http://www.homelandsecurity.org/journal/Articles/displayarticle.asp?article=50

10. Roger G. Johnston, "The 'Anti-Evidence' Approach to Tamper-Detection," *Packaging, Transport, Storage & Security of Radioactive Material* 16 (2005): 135–143.

11. The value of the anti-evidence in or on any given seal is secret, not the fact that anti-evidence is being used. Typically, the anti-evidence is changed to a new, unpredictable value when the seal is re-used.

12. The anti-evidence technique, in addition to being useful for seals, also has a number of important advantages for real-time ("live") monitoring and surveillance. This is called the "Town Crier" Method. It is discussed in Roger G. Johnston, Anthony R. E. Garcia, and Adam N. Pacheco, "The 'Town Crier' Approach to Monitoring," *International Journal of Radioactive Material Transport* 13 (2002): 117–126; and in Roger G. Johnston, Anthony R. E. Garcia, and Adam N. Pacheco, "Improved Security Via 'Town Crier' Monitoring," Proceedings of Waste Management '03, Tucson, AZ, February 24–27, 2003, www.osti.gov/servlets/purl/827636-nWiBFO/native/. If used in conjunction with OPODS, the Town Crier Method requires less reliable detection of physical or electronic tampering than conventional real-time monitoring approaches.

13. In the authors' experience, the common use of mechanical tamper switches on equipment or instrumentation cabinets to detect intrusion is approximately the same thing as having no tamper detection at all.

14. The authors are indebted to an anonymous reviewer for emphasizing the importance of clearly distinguishing between data integrity and data confidentiality. The former is the primary issue for this article, even though the authors are (unconventionally) proposing the use of a cipher for assuring data integrity.

15. Alfred J. Menezes, Paul C. van Oorschot, and Scott A. Vanstone, *Handbook of Applied Cryptography* (New York: CRC Press, 1996).

16. Douglas R. Stinson, *Cryptography: Theory and Practice* (Boca Raton, FL: CRC Press, 1995).

17. Bruce Schneier, *Applied Cryptography* (New York: Wiley, 1995).

18. It is sometimes said that, "Encryption does not provide data integrity." See, for example, Brad Conte, "Hashes," http://b-con.us/security/hashes.php. This statement is true, but only in a pedantic, formal sense. If an adversary is content to make random changes to the encrypted data without understanding the effect on the decrypted data (and no checksums or other hashes are included in the plaintext), then data encryption indeed cannot be used to reliably detect such random ciphertext tampering. However, in the case of nuclear monitoring, the goal of the nefarious adversary is to replace true monitoring data showing cheating with fake data showing nominal conditions. For example, if the facility operator wishes to sneak nuclear materials past a radiological portal monitor, he may want to replace the high radiation levels recorded by the monitoring equipment with normal background readings. Random vandalism to the encrypted data does not help him to accomplish this. Moreover, the plaintext data recorded with OPODS would almost certainly contain, in addition to the actual monitoring measurements, time stamps, measurement number, checksums, parity, or other simple hashes that would easily allow random data changing or rearranging to be detected.

19. Encryption or data authentication may also be used for communicating between two different people (or even oneself), separated in time, but at the same physically secure location. This is the typical scenario for securing computer data archives.

20. A typical statement made about hash or MAC security is that, "well, so far we haven't heard about anybody breaking it." This is not a very good metric for security, especially because governments who have broken a hash or MAC are not likely to publicly brag about it because doing so would limit their ability to exploit the discovery. With most modern ciphers, in contrast, mathematicians have a rigorous idea of the time and computational power required to break the cipher—at least using conventional methods. The problem, of course, is not knowing when or if a major cryptographic breakthrough might occur. With a one-time pad or OPODS, new cryptoanalytic methods in the future need not be feared because of the mathematically provable unbreakability.

21. Protechnix, "Cryptology and Data Security: The Vernam Cipher," http://www.protechnix.com/information/crypto/pages/vernam_base.html

22. See, for example, David A. McGrew and Scott R. Fluhrer, "Multiple Forgery Attacks Against Message Authentication Codes," http://eprint.iacr.org/2005/161.pdf; Bruce Schneier, "Schneier on Security," http://www.schneier.com/blog/archives/2005/02/sha1_broken.html; Bruce *Schneier*, *Secrets and Lies: Digital Security in a Networked World*. (New York: Wiley, 2000); Stefan Wolf, "Unconditional Security in Cryptography," in *Lectures on Data Security: Modern Cryptology in Theory and Practice*, ed. Ivan Damgard. (New York: Springer-Verlag, 1999), 217–250; Simon Singh, *The Code: The Evolution of Secrecy from Mary, Queen of Scots, to Quantum Cryptography* (New York: Doubleday,1999); Fred B. Wrixon, *Codes, Ciphers, & Other Cryptic & Clandestine Communication* (New York: Black Dog & Leventhal, 1998).

23. Sarah Granger, "Unlocking the Secrets of Crypto," http://www.securityfocus.com/infocus/1617

24. Although microprocessors can typically erase a single byte in 2 microseconds or less, erasing many bytes may take multiple milliseconds. Attacks on electronic circuits of less than a millisecond are quite possible based on making foreign connections to the electronic circuitry, disconnecting the power supply, or even by shooting the equivalent of a bullet through the correct location to break an electronic connection or a security bit. Moreover, microprocessors typically emit significant electromagnetic signals that are difficult to fully block. These emanations can be used to determine what the microprocessor is doing at a given instant, thus allowing an adversary to accurately time an attack, even at the sub-microsecond level.

25. It will generally be more difficult for the adversary to know the exact plaintext for monitoring measurements that are noisy or have a large dynamic range, but not for simple measurements such as whether the sensor did or did not detect a value above the alarm threshold.

26. Even if the adversary for international nuclear safeguards is not told the encryption or data authentication algorithm being used or given the microprocessor source code, he may know the algorithm being used, anyway, because Shannon's (Kerckhoffs') Maxim tells us an adversary can usually figure it out. See Roger G. Johnston, "Cryptography as a Model for Physical Security," *Journal of Security Administration* 24 (2001): 33–43; and Claude E Shannon, "Communication Theory of Secrecy Systems," *Bell System Technical Journal* 28 (1949): 656–715.

27. Gustavus J. Simmons, *Contemporary Cryptology: The Science of Information Integrity* (New York: Wiley-IEEE Press, 1999), 73ff; Bruce Schneier, *Applied Cryptography* (New York: Wiley, 1995), 17–18.

28. Bruce Schneier, John Kelsey, Doug Whiting, et al., "Performance Comparison of the AES Submissions," Proceedings of the Second AES Candidate Conference, NIST, March 1999, 15–34, http://www.schneier.com/paper-aes-performance.html

29. The severe budget constraints (see Ref. 7) imposed on the International Atomic Energy Agency (IAEA) increases the importance of using low-cost components in monitoring hardware.

30. See, for example, Stefan Wolf, "Unconditional Security in Cryptography," in *Lectures on Data Security: Modern Cryptology in Theory and Practice*, ed. Ivan Damgard (New York: Springer-Verlag, 1999), 217–250; and "One Time Pad" in *Glossary of Cryptographic Terms* (2000), http://www.pgp.net/pgpnet/pgp-faq/pgp-faq-glossary.html.

31. The Vernam cipher is called a one-time pad because the random key digits were originally written on a notepad. When all the digits on a given sheet were used up, the sheet was ripped off the notepad and destroyed.

32. David Kahn, *The Code-Breakers* (New York: Scribner, 1996). Lacking the one-time pad, captured encrypted messages have never been decrypted, and never will be.

33. The "modulus" or "wrap around function" is the remainder after integer division. Thus, for example, $9 \bmod 10 = 9$, $10 \bmod 10 = 0$, $11 \bmod 10 = 1$, $29 \bmod 10 = 9$. Similarly, $99 \bmod 100 = 99$, $100 \bmod 100 = 0$, $101 \bmod 99 = 1$, and so on.

34. A one-time pad can be used as part of a Machine Authentication Code (MAC) or to encrypt a hash. See, for example. Douglas R. Stinson, *Cryptography: Theory and Practice* (Boca Raton, FL: CRC Press, 1995). This does not, however, remove the problem of having to quickly erase many bytes should unauthorized access to the monitoring equipment be detected. Moreover, use of an encrypted hash or MAC based on a one-time pad does not guarantee that the adversary cannot discover fake plaintext data that covers up the treaty violation and results in an unchanged MAC tag or hash value. The latter is also a problem if OPODS were instead used in the MAC or to encrypt the hash.

35. Fred B. Wrixon, *Codes, Ciphers, & Other Cryptic & Clandestine Communication* (New York: Black Dog & Leventhal, 1998), 168–237.

36. Thus, the two 6s in the "1663" plaintext both encrypt to 0 in the substitution cipher's ciphertext "9003."

37. Substitution ciphers used to encrypt words can be easily broken using just a pencil and paper, plus a little working knowledge of the language being used. Only one or two sentences of ciphertext are ordinarily needed.

38. Like the Vernam cipher, OPODS is a stream cipher where the plaintext digits (or characters) are transformed independently of each other, in a time-varying manner. Unlike the Verman cipher, however, each ciphertext digit is related to its corresponding plaintext digit in a nontrivial, non-additive way.

39. In this example, if the adversary wanted to record fake monitoring data of "0000" instead of the true "1663" plaintext reading—indicating perhaps zero radiological counts instead of an elevated reading—the past erasure of OPODS mappings as they get used leaves him with an unsolvable dilemma. He knows that the first "1" in the plaintext maps to the ciphertext digit "2" via mapping #1 because he has access to the full ciphertext after breaking into the monitoring equipment, but the rest of mapping #1 is long gone. There is no information to help him decide what the plaintext digit "0" should map to. The problem is the same for the remaining 3 plaintext digits. And even if he guesses correctly for the first mapping, his guess is of no help in making guesses about the other 3 mappings. The use of some kind of checksum, parity, simple hash, time stamp, or measurement number that would also be encrypted using OPODS (along with the actual monitoring data) would further complicate an adversary's attempts to fake past or future data.

40. The odds of guessing the correct ciphertext decimal digit corresponding to a given plaintext decimal digit are 1 in 9, not 1 in 10. This is because the adversary already knows one of the digits in each mapping, assuming that he knows both the plaintext and the ciphertext from breaking surreptitiously into the monitoring hardware. (If, however, he only knows the ciphertext but not the plaintext, the odds are 1 in 10.) With a 1 in 9 chance of guessing the correct ciphertext digit, the odds of guessing correctly for even just 5 digits in a row are 1 in 59000. The use of hexadecimal digits (more practical for microprocessors) instead of decimal digits improves the odds to 1 in 15 for one digit, and 1 in 760000 for 5 digits in a row.

41. Hexadecimal (base 16) is the most practical and natural choice for microprocessors, and works no matter what the microprocessor word size because computers and microprocessors are currently binary devices, and will be for the foreseeable future. Each hex digit can be represented by 4 bits and belongs to the set 0,1,2,3,4,5,6,7,8,9,A,B,C,D,E,F. For example, hex A = binary 1010 = decimal 10, and hex F = binary 1111 = decimal 15. A single byte (8-bits) can be represented by two hex digits. Thus, decimal 256 = hex FF. If instead of using hex digits as the fundamental unit, bytes were used for OPODS, each mapping would 256 bytes long, which would require substantially more storage.

42. Having secure pre-trespassing data is useful even if post-trespassing data can be compromised because it constrains when attacks can take place. Typically, inspectors will be present at the start of monitoring, and for some time afterwards in order to verify the equipment is operating properly.

43. See, for example, James E. Gentle, *Random Number Generation and Monte Carlo Methods* (New York: Springer, 2003), 11–35; and William H. Press, Saul A. Teukolsky, William T. Vettering, and Brian P. Flannery, *Numerical Recipes in C: The Art of Scientific Programming* (Cambridge University Press, 1982), 274ff. For OPODS applications, this seed, as well as all subsequently computed pseudo-random numbers would probably be 2-bytes in length.

44. Note that Linear Congruent Generators (LCGs) are usually considered very poor choices for PRNGs and cryptographic security because it is often relatively easy to predict future results after examining a lot of their previous output. That is not of concern here because of the degeneracy (redundancy) built in (see Ref. 46), and because the adversary has access to only 0 or 1 data points (depending on where he interferes in the microprocessor cycle). The LCG cannot be predicted with so few data points even without the built-in degeneracy.

45. Instead of using OPODS, the authors' degenerate PRNG concept could be used to point to different MAC keys that get erased after each use. This would require less storage than OPODS, but (unlike OPODS) would be computationally intensive and could not guarantee the veracity of pre-trespassing data because there would always the possibility (no matter how difficult) that the adversary could generate fake data yielding the same MAC tag.

46. Should the current value of the PRNG get erased when trespassing is detected, but the adversary knows the PRNG algorithm plus manages to freeze the microprocessor in the middle of swapping or erasing a mapping, he may in principle be able to reconstruct the PRNG computations. This vulnerability can be substantially eliminated through use of "degeneracy," that is giving the PRNG output a higher precision than the number used to select the next mapping. For example, 2-byte (0–65535) PRNG values as proposed in this article, can be combined with the modulus 100 computation to determine which mapping (0–99) in the small cache of Figure 1 will be chosen next. That way, there are 655 different PRNG values on average that would result in choosing the same mapping. This redundancy means that the adversary can thus not reliably reconstruct the intended future PRNG values by simply knowing the current mapping, or which one would be chosen next. The authors have successfully demonstrated this degeneracy technique in there Time Trap anti-evidence seal: Roger G. Johnston, "Anti-Evidence Seals," Los Alamos National Laboratory Report LAUR-06-1312 (February 2006). In that application, there are an average of 400 different key values that produce the same hash result for a given time so that the adversary knowing the hash algorithm (but not the hash key) has only a 1 in 400 (0.25%) chance of correctly guessing the hash for future times.

47. An adversary concerned that the encrypted monitoring data shows that he has cheated, or concerned that he has botched a tampering attempt, can, of course, always erase the stored ciphertext, or lose, damage, or destroy the monitoring hardware, but the results of these actions will be noted by the inspectors.

48. Note that erasure of the PRNG algorithm itself is not necessary, as long as its current computation value has been erased. Being able to tell or even show the inspected nation the exact PRNG algorithm may have important transparency implications for international nuclear safeguards.

49. Guaranteeing the full erasure of information in memory or storage is not a trivial problem. See, for example, Peter Gutmann, "Data Remanence in Semiconductor Devices," http://www.cypherpunks.to/~peter/usenix01.pdf; Peter Gutmann, Proceedings of the Sixth USENIX Security Symposium, San Jose, CA, July 22–25, 1996, http://www.cs.aukland.ac.nz/%7Epgut001/pubs/secure_del.html.

50. The faster the erasure, the less likely an adversary can interrupt it, the more chances the microprocessor has to overwrite memory multiple times to remove data remanence, and the better the security. It is preferable to erase faster than the maximum practical mechanical speed: the time for sound to cross a microchip 5 mm in length, which is ~15 microseconds. The OPODS 2-byte PRNG value can be erased from microprocessor RAM in 2 microseconds or less. In contrast, erasing a 256-byte key (which often will not fit in microprocessor RAM) from EEPROM typically takes over 120 microseconds.

51. The mappings cannot, with good security, be generated as needed by a deterministic pseudo-random number generator (PRNG) built into the monitoring hardware. Even if the current PRNG value can be erased immediately after trespassing is detected—something that cannot be guaranteed—an adversary would have enough information from the PRNG algorithm, plus the extensive plaintext and recorded ciphertext, to easily figure out past and future mappings. If the PRNG algorithm could be kept secret and reliably erased—something not compatible with transparency or host national security—a sophisticated adversary would still be able to figure out the mappings, especially if large amounts of ciphertext were available. It is not, of course, possible to generate the mappings as they are needed from a non-deterministic hardware random number generator built into the monitoring hardware because the mappings would then not be known to the inspectors.

52. One OPODS mapping is required for each plaintext hex digit to be encrypted, and two mappings for each plaintext byte. Complex compression algorithms such as zip compression can further reduce the storage required for OPODS mappings, typically by 5–10%, but greatly increase the computational complexity of decompression.

53. Compiled partially from "Historical Notes about the Cost of Hard Disk Storage Space," file:///Volumes/Corsair%202GB/Cost%20of%20Hard%20Drive%20Space.webarchive and Thomas M. Coughlin, "Flash Illuminates Mobile Digital Storage," http://www.entertainmentstorage.org/articles/A%20storage%20vision%202.pdf

54. microEngineering Labs, http://www.melabs.com

55. Bruce Schneier, "Twofish: A 128-Bit Block Cipher," http://www.schneier.com/paper-twofish-paper.html

56. The great mathematician John von Neumann (1903–1957) once remarked that, "Anyone who attempts to generate random numbers by deterministic means is, of course, living in a state of sin." Quoted in Herman H. Goldstine, *The Computer from Pascal to von Neumann* (Princeton, NJ: Princeton University Press, 1972), 378.

57. William H. Press, Saul A. Teukolsky, William T. Vettering, and Brian P. Flannery, *Numerical Recipes in C: The Art of Scientific Programming* (Cambridge University Press, 1982), 274ff.

58. Kevin D. Mitnick and William L. Simon, *The Art of Intrusion: The Real Stories Behind the Exploits of Hackers, Intruders, & Deceivers* (New York: Wiley, 2005); J. Kelsey, B. Schneier, D. Wagner, and C. Hall, "Cryptanalytic Attacks on Pseudorandom Number Generators," Fast Software Encryption, Fifth International Workshop Proceedings, March 1998, Springer-Verlag, 168–188, http://www.schneier.com/paper-prngs.pdf; Peter Gutmann, "Software Generation of Practically Strong Random Numbers," https://www.usenix.org/publications/library/proceedings/sec98/summaries/

59. The 16th step is not required if the "List 15 Hex Digits Algorithm" is used to store the mappings in the microprocessor used for monitoring in the field. The microprocessor can figure out the final hex digit in each mapping because it is the only one of the 16 hex digits not yet appearing in the mapping.

60. Because each mapping ultimately depends on the random list of hex digits generated non-deterministically by hardware (not a PRNG), the fact that the "unused list" is scrambled pseudo-randomly by computer should not greatly compromise security.

61. Paul E. Black, "Fisher-Yates shuffle," in Dictionary of Algorithms and Data Structures [online], Paul E. Black, ed., U.S. National Institute of Standards and Technology. December 19, 2005, http://www.nist.gov/dads/HTML/fisherYatesShuffle.html. Note that 16 shuffles, for example, requires 32 OTP digits, so Fisher-Yates (Knuth) Shuffling does not make very efficient use of the OTP.

62. Generating the OPODS mapping, of course, is done at headquarters prior to running the monitoring hardware so computation time is not as critical as if it had to be done in real-time by the monitoring microprocessor in the field.

63. REALbasic is an object oriented, cross-platform compiled language and programming environment developed by Real Software, http://realsoftware.com.

64. The completeness of quantum mechanics seems to suggest that not even the universe cannot predict random numbers generated by quantum effects.

65. See, for example, John Walker, "HotBits: Genuine Random Numbers, Generated by Radioactive Decay," http://www.fourmilab.ch/hotbits/; Black Cat Systems, "Generating Random Numbers Using Radiation," http://www.blackcatsystems.com/GM/random.html; Wired Online, "Totally Random," http://www.wired.com/wired/archive/11.08/random_pr.html

66. For examples of commercial electronic PRNGs, see Orion Products, "Random Number Generator," http://www.randomnumbergenerator.nl; ComScire, "Design Principles and Testing of the QNG Mode J1000KU," http://comscire.com/Products/J1000KU/; "Hardware Random Bit Generator," http://willware.net:8080/hw-rng.html; and Protego, "SG100 TRNG," http://www.protego.se/sg100_en.htm

67. See, for example, Quantique, "Quantum Random Number Generators," http://www.idquantique.com/products/quantis.htm; Thomas Jennewein, Ulrich Achleitner, Gregor Weihs, Harald Weinfurter, and Anto Zellinger, "A Fast and Compact Quantum Random Number Generator," *Review of Scientific Instruments* 71 (2000): 1675–1680; Ma Hai-Qiang, Wang Su-Mei, Zhang Da, Chang Jun-Tao, Ji Ling-Ling, Hou Yan-Xue, and Wu Ling-An, "A Random Number Generator Based on Quantum Entangled Photon Pairs," *Chinese Physics Letters* 21 (2004): 1961–1964, http://www.ingentaconnect.com/content/iop/cpl/2004/00000021/00000010/art00027.

68. Mahesh Joharl, "Really Random Numbers," http://www.dirfile.com/really_random_numbers.htm.

69. Wired Online, "Totally Random," http://www.wired.com/wired/archive/11.08/random_pr.html.

70. Don Davis, Ross Ihaka, and Philip Fenstermacher, "Cryptographic Randomness from Air Turbulence in Disk Drives," http://world.std.com/~dtd/random/forward.pdf.

71. Richard P. Dunnigan, "Random Number Generator," U.S. Patent 4,786,056.

72. A true random sequence cannot be generated based on English language text. See, for example, Claude E Shannon, "Prediction and Entropy of Printed English," *Bell System Technical Journal* 30 (1951): 50–64 because of the patterns inherent in any language. Generating random or pseudo-random sequences from human use of a computer keyboard or mouse is based on the exact microseconds when a key is pressed or the mouse is clicked. The authors have used a similar technique in many of their anti-evidence seals. See Roger G. Johnston, "Anti-Evidence Seals," Los Alamos National Laboratory Report LAUR-06-1312, February 2006. The human action randomly or pseudo-randomly determines when a PRNG should stop generating pseudo-random numbers; the PRNG value when stopped is chosen as the pseudo-random number to use.

73. Eric R. Gerdes, Roger G. Johnston, and James E. Doyle, "A Proposed Approach for Monitoring Nuclear Warhead Dismantlement," *Science & Global Security* 9 (2001): 113–141.

74. Data compression allows even more storage.

75. Bruce Schneier, *Crypto-Gram Newsletter*, June 15, 1998, http://www.schneier.com/crypto-gram-9806.html

76. In the authors' view, 3rd party authentication of international safeguards data is of marginal value, anyway. A 3rd party that is dubious of safeguards findings will not be (nor should they be) convinced of truthfulness based only on the apparent integrity of stored or transmitted data. The possibility that monitoring hardware has been tampered with or has malfunctioned, or that the measurements have simply been misinterpreted, will always be concerns that are extremely difficult to assuage—especially in the minds of parties who do not want psychologically to admit to the possibility of treaty breakout.