Terakey

An Encryption Method Whose Security Can Be Analyzed from First Principles

Arnold G. Reinhold
Cambridge, Massachusetts
July 3, 2020

Rev 1.0

Abstract

Terakey™ is an encryption system whose confidentiality can be demonstrated from first principles, without making assumptions about the computational difficulty of certain mathematical problems. It employs a key that is much larger than the anticipated volume of message traffic. It is based on the one-time pad, but addresses the risk of key reuse stochastically. Conventional cryptographic techniques can be used to ameliorate infrequent byte collisions. The large size of the key reduces the risk of key exfiltration and facilitates physical security measures to maintain a secure chain of control for the key. Terakey also serves as a potential alternative for comparison with quantum key distribution technology, arguably providing equivalent security with fewer complications.

Introduction

Terakey™ is an encryption system based on a shared secret key that is much larger than the anticipated volume of message traffic. Its intrinsic security can be demonstrated from first principles, without making assumptions about the computational difficulty of mathematical problems, such as factoring large integers or computing logarithms in finite groups.

Terakey extends the provable security of the one-time pad (OTP) to network operation. A central requirement for one-time pad security is that pad data never be used more than once. This requirement has largely limited traditional one-time pads to protecting either a single pair of communicators or a star network topology where a central station decrypts each message from an originator and re-encrypts it for the destination. Terakey attempts to satisfy the no-reuse requirement stochastically, by using a very large unpredictable secret, the Terakey, that is shared with all stations. Any pair of stations can generate a unique one-time pad from the Terakey for each message. There is a risk of collisions, where Terakey bytes are used more than once. The risk is quantifiable and can be kept low by keeping the Terakey much larger than the volume of traffic. A variety of

approaches are presented below to minimize damage from the infrequent collisions that may still occur.

Terakey's large key size has an additional advantage of making theft of the key more difficult. While designers of existing encryption systems generally consider short keys a virtue, that assumption may not be valid in all circumstances. Lightweight keys, a few thousand bits or less, can be surreptitiously purloined via low-bandwidth side channels. A heavy-weight Terakey makes such attacks much less feasible. Terakey size can be chosen to require minimum physical dimensions and mass for its storage, based on current and projected memory technology. Large material objects are more easily protected by physical security. The Terakey can be stored in an isolated module within a guarded and alarmed security safe. A specialized security processor can be used to verify the format of requests for key data and restrict the rate at which data is retrieved from the Terakey storage device, to further reduce the risk of key exfiltration.

Terakey is designed for applications where prior secure physical transport and storage of the key file is feasible, though keys can be refreshed remotely. Examples might be communication between fixed sites, such as financial institutions with multiple branches, or between mobile facilities of sufficient size, such as ships or aircraft. Terakey can be used to protect moderate sized message traffic, such as text-only electronic mail, or used to exchange traffic encryption session keys for use with a conventional cipher, such as AES. In the latter case, of course, security will also depend on the strength of the chosen session cipher.

Terakey systems can be constructed using off the shelf electronic hardware components and relatively simple software. The rapid drop in the cost per terabyte for mass storage, and for solid state flash storage in particular, helps make Terakey attractive.

Terakey can also serve as a theoretical test case for evaluating quantum key distribution technology (QKD) [1]. Its session key exchange functionality parallels the capability of QKD, making Terakey an alternative for comparison, arguably offering equivalent benefits with lower cost and risk. Because Terakey's security derives from first principles, it is inherently resistant to cryptanalytic attack by quantum computers, a potential threat to existing cryptographic algorithms, for which solutions are still in development [2].

Table of Contents

Security limitations

In Terakey's simplest form, there is a risk of occasional message bytes being compromised. A central assumption for the security proof of one-time pads is that key must never be reused. Terakey does not include a mechanism to completely prevent reuse, but instead minimizes its probability. The risk of key byte reuse increases as the ratio of traffic sent using a given Terakey to the size of that Terakey. By keeping the quantity of key high compared to the cumulative volume of traffic, collisions resulting in key byte reuse can be made infrequent.

Key byte reuse is most problematic if an adversary acquires the plaintext of the previous traffic encrypted by that byte, in which case they can recover the plaintext encrypted the second time with that key byte. To exploit byte collisions in this way, an adversary needs a substantial fraction of previously sent plaintext. While it may be difficult for an adversary to accumulate large quantities of the plaintext traffic on a network, it is generally unwise to base security on such an assumption, as demonstrated by the massive leak of classified U.S. diplomatic cables to Wikileaks in 2010. There are several possible ways to deal with the impact of low-probability byte collisions:

o Living with the risk. An adversary knowing a byte or two in a text message may not be that damaging in practice.

o Using more than one round of Terakey, which greatly reduces the probability of collisions, as described below.

o Using classical techniques, such as random message padding or message folding, to obscure the occasional collision.

o Using a modern cryptographic pseudorandom number generator to shield the key-byte selection process. This last approach is simple and natural to do, but the resulting protection against byte collisions is not proved from first principle. However, the security assumptions for the cryptographic tools used for this purpose can be less stringent than for conventional usage. Recovering the state of a cryptographic pseudo-random number generator from occasional bytes of output is presumably much harder that recovering the state if large quantities of previous output is known.

o Using Terakey exclusively to exchange symmetric keys. An adversary would get limited benefit from knowing a byte or two of a 256-bit key. More importantly, if a Terakey is only used for key exchange, discovering byte reuse requires the attacker to recover a large fraction of the session keys employed to encrypt previous messages (a known session key attack). Knowing just the plaintext of the messages does not enable detecting byte reuse. Assuming the symmetric cipher used has significant strength, recovering numerous session keys would be difficult even if all previous message plaintexts are known. This approach provides a basis for comparison to quantum key distribution, as discussed below.

Background

Modern encryption methods convert data between plain text and cipher text under the control of a block of data called a *key*. Security depends keeping that key secret from everyone but the persons or devices authorized to access the data. Asymmetric or public key encryption systems employ two keys, one that is published for use in encrypting data and a private key for decrypting it. For security, the private key must be kept secret from unauthorized entities.

Traditional encryption methods can be divided into two classes based on the size of the key:

o ciphers, where the secret key is much smaller than typical plaintext. This category includes both symmetric and asymmetric encryption methods.

o the one-time pad (OTP), where the key used to encrypt a message is equal in size to the plaintext.

Some ciphers, such as AES, are highly regarded, but to date there is no formal proof of their security, at least not in the public record. Published attacks on RC4, MD5 and SHA1 raise some (perhaps small) doubt as to the long-term invulnerability of any specific cipher algorithm. Quantum computers could attack some asymmetric ciphers and reduce the strength of symmetric ciphers. In addition, systems that are based on relatively short keys are subject to a variety of side channel attacks, including the recent exploits of the speculative execution features of modern processors, without which processor performance degrades. The one-time pad, however, when constructed correctly and used properly, is provably secure [3].

Terakey is a third encryption category that employs a secret key that is much larger than the plaintext. At first glance this seems pointless. If the one-time pad offers complete security with a key no longer than the plaintext, why use a much larger key?

The answer is to fix a major shortcoming of OTP. For full security, no portion of a key can ever be used twice. This means that each party using OTP must keep careful records of which pads have been used. When only two stations are communicating this is burdensome enough. But when multiple stations wish to communicate with each other directly, without going through a central relay station, the bookkeeping requirements quickly become unmanageable. The number of station pairs, and hence the number of sets of pads to be managed, increases as the square of the number of stations.

Terakey eliminates the need to manage numerous pads. The primary trade-off is security that can degrade slowly as the volume of traffic increases, but in a quantifiable and manageable way. Assuming an adversary has copies of plain and ciphertext from previous transmissions and all details of the encryption are known except for the Terakey itself, there is a small probability that the adversary can recover some information about an occasional byte in a message. Typically, this would be the *xor* of the two bytes (depth

2). Higher depth collisions are possible but much rarer. If the adversary knows or can guess the plain text of one of the messages, they can read the corresponding character in a new message. The probability of such a byte collision increases gradually with the amount of traffic encrypted using a given Terakey. However, the probability can be made small in practical circumstances.

Using very large random files for security has previously been proposed for deniable file systems by Assange, Dreyfus and Weinmann, [4] and by Spilman and us for securing password validation data. [5] [6]

Description of Terakey

To understand its strengths and limitations, Terakey will be described in a series of versions or levels. For full security, a higher level would presumably be used, however, to analyze its properties, we start with the simplest version, which best illustrates the byte collision problem, and then describe other versions that address the byte collision problem in various ways.

Terakey consists of three major elements:

1. A large master key string, K of length *L* bytes, the Terakey. Bytes would typically be eight bits, for ease of use with common hardware, but could be any fixed number of bits. *L* is assumed to be much larger than the typical network traffic volume during the time *K* is in effect: gigabyte, terabyte or even petabyte keys might be appropriate.
2. A method for assigning each message a unique message indicator *v,* which can be determined by a transmitted nonce, by hashing the message, by hashing just its header, from a prearranged list, communicated using a separate mechanism, such as conventional public key cryptography, or some combination of these methods.
3. A deterministic pseudorandom number generator function, R(*v, n*), whose output ranges from 0 to *L*-1. The message indicator *v* is used as the control variable, or seed, for R, i.e. R produces different pseudorandom sequences for different *v.* The variable *n* indexes the bytes in the pseudorandom sequence produced by R. The index *n* starts at 0 for each new message.

For our security analysis, we assume that an attacker knows all the design details of Terakey and what options are in use. For now, we further assume that the attacker also knows both R and *v,* can infer *n,* but does not know K

Terakey basic

In Terakey basic, the n^{th} plaintext byte of a message, P(*n*), is converted to cipher text, C, by the formula C(*n*) = P(*n*) o K(R(*v, n*)), where *v* is the message indicator and o is a reversible operation such as bit-wise exclusive-or *(xor)* or modular addition. For decryption, P(*n*) = K(R(*v, n*)) o' C(*n*), where o' is the inverse of o. (For *xor,* o' is the same as o.)

If an attacker knows the plaintext and cipher test of previous messages, then he can recover the Terakey bytes that were used to encrypt those messages. If a new message happens to use one of those recovered key bytes in the k^{th} position, then k^{th} byte of the new message will be compromised, a byte collision.

Let M be the total number of bytes of message plaintext that has been obtained by an adversary (worst case, the total amount of traffic that has been sent to date on the network using the current K). Let $f = M/L$ be the relative amount, i.e. the fraction of the Terakey that has been used. Then if $M << L$, the probability that a byte in a new message can be read by an attacker who knows the plaintexts of past traffic is f. For example, if $L = 10^{12}$ (1 TB) and M= 4 GB, then f = .004. So, in a 1000-byte message, an average of 4 bytes will be potentially compromised.

Even if an attacker does not know the plaintexts of past traffic, he may, under the basic assumptions, be able to discover instances where the same key byte was used to encrypt two different plaintext bytes, *p1* and *p2.* The attacker will then know the value of the operation o' applied to the two plaintext bytes: $(p1 \text{ o } k)$ o' $(p2 \text{ o } k) = p1$ o' $p2$. Classically, when an attacker knows such byte collisions for a large block of contiguous characters representing English, or other natural languages, it is often possible to recover both plaintexts, due to redundancy in written languages. A few sporadic byte collisions would be less vulnerable.

Any message byte encrypted by a Terakey byte that has never been previously used on the network is secure, however a byte that is secure when sent can be compromised by future traffic. Terakey basic is imperfect, but is strengths and weaknesses are completely characterized from first principles. The goal is to build on basic's strengths and minimize its weaknesses.

Terakey doubled

The likelihood of a byte collisions can be sharply reduced by using two Terakey bytes to encrypt each plaintext byte. In Terakey doubled, the nth plaintext byte P(n) is converted to ciphertext by the formula:

$$C(n) = P(n) \text{ o } K(R(2n)) \text{ o } K(R(2n+1))$$

(Here we omit the seed parameter v in R for clarity.) An adversary who knows C(n) and P(n) only learns K(R(2n)) o K(R(2n+*1*)). Assuming f is small, the probability of adversary encountering K(R(2n)) o K(R(2n+*1*)) in the encryption of another cipher byte is roughly the square of 2f.

With Terakey doubled **t**here are, however, other situations where a new cipherbyte can be compromised. Suppose an attacker, who we are still assuming knows past traffic and how the bytes of K are selected, has recovered, in the past, messages containing the first

two cipherbyte pairs below, and therefore knows C1, C2, P1, P2, K(R(*a*)) o K(R(b)), and K(R(b)) o K(R(c)).

C1 = P1 o K(R(*a*)) o K(R(*b*))
C2 = P2 o K(R(*b*)) o K(R(*c*))

The attacker now intercepts a new secret message

C3 = P3 o K(R(*a*)) o K(R(c))

and therefore, knows C3, and that R(*a*) and R(*b*) are being used to encipher P3. She can then recover P3 by computing

(K(R(*a*)) o K(R(*b*))) o (K(R(*b*)) o K(R(*c*))) = K(R(*a*)) o K(R(*c*))

Note that this scenario involves three coincidences, one for C2 and two for C3, and so its likelihood is on the order of f^3 for small f. Longer chains are also possible but even less likely.

Again, assume L = 10^{12} (1 TB) and M= 4 GB, Then f = .004 and $(2f)^2 = 6.4 * 10^{-5}$. So, in a 1000-byte message the probability of one byte being compromised is $6.4 * 10^{-2}$.

The penalty, of course, is roughly doubling the time required to extract the session key from the Terakey key file and touching twice as many key bytes per message.

Terakey m-level

The likelihood of a byte disclosure can be further reduced be using more than two key bytes per plaintext bytes. In level *m*, the *n*th plaintext byte P(*n*) is converted to ciphertext by the formula:

C(*n*) = P(n) o K(R(2*n*)) o K(R(*mn*+*1*)) o … o K(R(*mn*+*m*-*1*))

Again, as long as *f* is small, then the probability compromised bytes at level *m* is on the order of $(mf)^m$.

Terakey Indirect: Use one key byte to help select the next

Another way likelihood of a byte disclosure can be sharply reduced using two Terakey bytes to encrypt each plaintext byte is to use one byte to help select the second bye. Here the *n*th plaintext byte P(*n*) is converted to ciphertext by the formula:

C(*n*) = P(*n*) o K((R(2*n*) + K(R(2*n*+*1*)) mod L)

or, alternatively:

$$C(n) = P(n) \text{ o } K((R(n) + K(R(n)) \bmod L)$$

Terakey SKRNG: Use a secretly keyed strong random number generator for R

In our analysis of Terakey security so far, we assumed that the algorithm R, its seed and therefore its outputs are known to an attacker. We did this to analyze the byte collision phenomenon under worst-case conditions, so as to establish its confidentiality characteristics from first principles. In practice there is no reason to freely expose the state and output of R to potential attackers. We can use a keyed cryptographic random number generator for R, and seed it in one of a variety of ways:

1. Instead of using a message indicator to seed R, we can use that value to look up a string in K and then use that secret string to seed R. This might be done using the basic Terakey approach, that is first using the message indicator to seed R, then using R to select enough bytes to reseed R.
2. We can separately provide a secret list of seeds for R, changed daily or after some other crypto period. If the seeds are destroyed after use, the damage from an attacker capturing K would be limited
3. Use public key cryptography to exchange seeds for R on a per message basis. This would allow Terakey to benefit from the forward security properties of the public key system and allow pairs or groups of Terakey key holders to communicate with privacy from other holders of that Terakey key.
4. Of course, a combination of the above methods could be used.

Additional measurers to deal with byte collisions

Below are several additional ways to deal with byte collisions that apply to any Terakey level:

Using Terakey exclusively to exchange session keys

As pointed out above, another use case where occasional byte collisions might be acceptable is when Terakey is used to exchange session cipher keys. Knowing two bytes in a cipher key reduces the strength of, say, a 256-bit key to 240 bits, which is still strong. And if a Terakey is only used to exchange cipher keys, a byte collision attack on a Terakey requires an attacker recovering a substantial fraction of the session cipher keys, not just the messages sent on those keys. This is a much more challenging task, even if all message plaintexts sent to date are known, as it requires breaking the cipher to recover session keys for a large fraction of messages.

Furthermore, the volume of traffic when only cipher keys are exchanged would be much lower. A 256-bit key is only 32 bytes long. A station generating one new 256-bit key per second would consume just over one gigabyte of Terakey per year. Restricting Terakey to

exchanging session keys also reduces the likelihood of byte collisions. In the example given for basic Terakey above where $L = 10^{12}$ (1 TB) and M= 4 GB (four years at one session key per second, with all session keys recovered by the adversary), then $f = .004$. and the probability of a single byte collision in a 256-bit key (32-bytes) would be 0.128, the probability of a two-byte collision 0.0164, three bytes 0.0021. An eight-byte compromise would occur about every 13.8 million keys. Reducing f, either by less volume or a bigger Terakey, would reduce these small risks dramatically.

The short length of symmetric keys forces an attacker to compromise a larger number of key exchange messages than would be needed if text messages were being encrypted directly and their plain text was recovered. If any material risk of leaking bytes from session keys is considered unacceptable, Terakey could be used to create a string longer than the required cipher key with that string then hashed to form the traffic key. This would further reduce the value of knowing even a few bytes of the pre-key string.

Change K periodically

Changing *K* periodically helps keep *f* small. A new K can be supplied regularly, or *K* can be stirred periodically using some pseudo random function, perhaps keyed from *K* based on a seed exchanged physically or by public key methods, or from a fresh data set, perhaps on DVD-ROMs, delivered to each site, or both. Backup Terakeys can be kept on hand allow a fresh *K* to be employed in a crisis. Terakeys can be updated by sending fresh Terakeys, which can be used as-is or combined with the existing Terakey using the *xor* function or a more complex algorithm. The new key might be placed in service when all stations confirm successful update.

Use random message padding or message folding

Using classical techniques, such as adding random strings at the beginning and end of a message or dividing a message at a random point and sending the second half first would obscure the location within the message of occasional byte collisions. The former method also disguises the message size, which could otherwise reveal information useful to an attacker.

Pre-encrypt the message using a standard cipher

Terakey could be used to super-encrypt messages protected by conventional ciphers. Security becomes dependent on assumptions that are much more relaxed than normal security assumptions for the cipher alone and may be fully provable. For example, if f is small enough so that the probability of more than a single byte collision is low, then even pre-encrypting the message with a Caesar cipher with a random shift selected from K could eliminate any information from a single byte collision.

Authentication

Terakey is a stream cipher and therefore subject to malleability attacks. Someone who knows the content of a message in transit can alter the message so that it decrypts to an altered form. Random padding or folding complicates such attacks. Standard cryptographic techniques such as message authentication codes (MACs) can be employed, potentially using the Terakey to supply the needed shared secret.

Forward Security

Because destruction of a Terakey can be difficult, there is a risk of the key being compromised by a physical attack. The adversary who recovers the Terakey intact might then be able to decode past messages encrypted using that edition of the Terakey. One way to prevent this is to exchange message indicators using public key cryptography algorithms that provide forward security. The security of this approach relies on the presumed security of the public key algorithm employed. However, standard attacks on pubic key systems, including quantum attacks, assume the attacker knows the public key and can attempt to break it at leisure, e.g. by factoring an RSA public key. With Terakey there is no reason to offer the attacker that luxury. Stations or users can keep their public keys secret from third parties, exchanging them encrypted with Terakey. As long as these not-so-public keys are exchanged infrequently, the additional traffic burden on the Terakey would be negligible. This approach to exchanging message indicators also permits compartmentalization: pairs or groups of Terakey key holders wishing to communicate with privacy from other holders of that Terakey key edition can share public keys only within those groups.

Logistics

A Terakey can be stored on a single hard drive, multiple hard drives, RAID array, one or more solid state drives, SD card, USB drives, CD, DVD, or Blu-ray optical store, or any other random-access electronic storage medium. Keys can be transported disguised as audio or video discs or in mobile devices, such as thumb drives, smart phones, music players or tablets. Two or more Terakeys can be generated separately, sent via separate channels, e.g. two couriers, and combined on arrival.

Terakey users in a network could track the quantity of Terakey used and periodically report approximate volume to a central point to estimate the overall fraction f of Terakey that has been used and the rate at which it is being used. Replenishment of the Terakey can be based on these reports.

Terakey can be housed in sealed box or safe with, say, optically coupled I/O with limited bandwidth. The enclosure could be equipped with alarms that would trigger data erasure if unauthorized tampering were detected. All communication with the Terakey storage module can be routed through a specialized security processor that verifies the format of

requests for key data, to prevent attacks by malformed requests. The security processor could also restrict the rate at which data is retrieved from the storage device, to reduce the risk of large parts of the key being stolen, and signal an alarm if excessive requests were detected.

Key generation

The full security proof for Terakey assumes that the entire Terakey is random. Creating a Terakey therefore requires large quantities of fully random data. While the needed engineering is not trivial, this article assumes it is possible to generate enough such data of sufficient quality to assure security.

High speed video digitizers with 12 to 14 bit resolution and speeds over 200 million samples per second are commercially available [7], as are calibrated broadband noise sources [8]. A noise source constructed from readily available components [9] may be suitable if it is desired to minimize the number of external devices that need to be trusted. A commercial and bespoke noise sources could be digitized separately and combined. NIST Special Publication 800-90B [10] contains validation and health test procedures for nose sources intended for use in cryptographic equipment. If entropy can be generated at 100 MB/s, a 2 TB SSD memory could be filled in under 6 hours.

As a practical matter, a combination of a wide-band random data source and data whitening techniques could be used. Intermediate solutions that rely on cryptographic primitives, such as ciphers and hashes, seeded with sufficient entropy can be used to generate a Terakey, as suggested in NIST SP800-90 [11], gaining the benefit of heavyweight encryption, but without the first-principles Terakey security proof.

Key destruction

Destruction of a Terakey, say when physical security is being threatened, is more difficult than for a traditional short key. The classic approach, thermite [12], would violate most current workplace safety standards and fire codes. Writing over data on hard disks can be used to sanitize them, but the process can take hours for large capacity drives. NAND solid state drive technology allows for rapid erase function by setting all bits in a block to ones. and many solid state drives have a special ATA Secure Erase command that perform this function on all blocks [13]. The Secure Erase command takes time proportional to the drive's capacity, one vendor's website estimates 20 seconds for a 512 GB unit [14]. Storing the Terakey in a physical safe, with an internal back-up battery, would allow completion of the Secure Erase function, even in the event external power was removed in an active attack.

Another approach would be to use whole disk encryption to protect the Terakey, with provision for destroying the disk encryption key when necessary. NIST calls this method Cryptographic Erase [15]. Note that this is no less effective than similar measures commonly employed to protect decrypted messages and other sensitive information in storage, and also could protect the Terakey during transport, assuming the disk key is sent

separately. Cryptographic erase is nearly instantaneous, but auditing devices to insure it is being done properly is difficult. Cryptographic erase might be combined with ATA Secure Erase.

Overwriting is not generally recommended as a way to sanitize SSDs because the drive controller manages the relationship between logical addresses and flash memory physical addresses to reduce wear on the flash memory, which tolerates a limited number of write cycles per block. This can result in sensitive information being stored in inactive blocks that are no longer accessible. However, for Terakey, the SSD is written to infrequently, at initialization and during any refresh, so the fragmentation from wear leveling may not occur. SSDs have faster write speeds (3,300 MB/s) [16] than rotating hard drives, but even then, overwriting a two terabyte SSD module could take ten minutes. Overwrite could serve as a fall back if the Secure Erase command fails, as reflected by a read test that did not return all ones. Terakey is somewhat resistant to partial key recovery since key bytes from many random sectors of the Terakey are required for each encryption or decryption.

Physical destruction is the ultimate approach. The United States National Security Agency publishes guidelines for destroying different types of digital media, along with lists of approved destruction device vendors [17].

Resilience

Terakey is intrinsically resilient to storage failures. Bad blocks in the key store will only be used infrequently. If detected on transmission, a new seed for R can be generated, which will be unlikely to need the bad block, or error-correcting codes can be used to recover from the occasional garbled byte. Even if an entire solid-state memory unit fails and there is no loaded spare, other stations can be advised and can fall back to a Terakey that excludes the data in the bad module when communicating with the station that experienced the failure. Provisions for this situation can be included in the Terakey software. Each loaded Terakey storage module could have a unique ID, perhaps derived from a hash of its contents. The stations on the network could maintain tables of the modules online at each location, and form Terakeys from only the modules available at the destinations.

Performance

Storing the Terakey in RAM would offer the highest performance. Solid state drives would offer the next best performance. Solid state drives currently cost about twice as much per gigabyte of storage as rotating magnetic disks, but the overall cost per Terabyte is low enough that rotating media would likely only be used if there were specific reasons to prefer them.

If rotating media are used to store K, message encryption and decryption will typically require one disk access per byte of message. RAID technology would spread the Terakey over multiple disks, which can speed access. Speed can be further improved by creating a list of the bytes from K that will be required and sorting them by disk, track and sector order. Then there will typically be 1/2 disk rotation latency per byte of message. At 7200 RPM, a 1200-byte message would require about 5 seconds to encrypt or decrypt on a single hard drive. About seven 256-bit session keys could be processed per second. Multiple messages can be batched, their keys sorted and merged for access and then decrypted together to save time. All the computations can be pre-computed before the message arrives, if the nonce is known, to produce a set of working keys of some fixed size. Nonces could be pre-agreed, say a hash of the date, sending station name, receiving station name, message number, and, perhaps, a secret of the day, sent the previous day. These precomputed working keys could then be used to rapidly decode messages.

Comparison with Quantum Key Distribution systems

Terakey is a possible alternative for comparison with quantum cryptography as a way to exchange traffic encryption keys. Terakey is arguably superior in terms of cost and auditability, while providing comparable confidentiality on a first-principle proven basis.

Communication requirements

Unlike QKD, Terakey has no requirement to maintain quantum coherence over long distance. A dedicated fiber channel is required for many versions of QKD. Not only is this a cost issue, but also it makes QKD more subject to denial of service attack. Terakey can use any communication channel, including the public Internet, dedicated lines, HF radio, and even paper messages. There is also no need for a backchannel to coordinate quantum measurements, as in the BB84 protocol, for example. The receiving party in a Terakey communication can maintain complete radio silence, which can be important for applications such as naval fleet broadcast or communicating with non-governmental organizations in chaotic regions.

Cost

Terakey can be implemented on any computer with a mass storage port. The cost for solid state storage is under $120 per terabyte as of July 2020. Price quotations for off-the-shelf Quantum Key Distribution (QKD) systems are hard to come by.

Quantum-derived security

QKD cleverly employs the entanglement phenomenon that Albert Einstein called “spooky action at a distance” to create matching keys at two locations without possibility of interception. The randomness of those keys, however, depends on quantum indeterminacy, the unpredictability of exact outcomes of events at the quantum level. The

same quantum indeterminacy makes the electronic noise sources that would be used in creating a Terakey just as unpredictable.

Auditability

Terakey systems can be built from off-the-shelf mass-market components with relatively simple software that can be understood by ordinary software engineers and audited with conventional software configuration control technology. QKD systems are only available from a small number of vendors located in a few countries. Trusting QKD involves trusting those vendors and the governments where they operate. Auditing QKD requires very specialized knowledge, at the PhD in quantum physics level, assuming QKD providers are willing to allow an audit. Few potential user organizations have the technical depth to audit QKD systems.

Maintainability

Terakey uses standard, off-the-shelf components. Spare parts can be procured over the counter locally. Full back up is inexpensive. Maintenance can be performed by ordinary computer technicians. QKD systems use sensitive optical components and typically require cryogenic refrigeration. Specialized training is required to maintain QKD systems and significant investment is backup systems and spares would be required to insure availability at remote locations. Unless ample stocks of spare parts were supplied with each QKD system, additional spare parts would have to be obtained from the manufacturer and shipped securely.

Need for key refresh

QKD by its nature does not need key refresh. At the slow speeds of current QKD systems, an unrefreshed Terakey could be usable for years. Terakey's security would degrade over time under the assumption that an attacker knows all or a large fraction of previous messages sent, the "known plaintext attack." However, in the key distribution use case "known plaintext" actually means knowing the session key for each message sent. So, for example, if Terakey is used to exchange AES session keys for individual emails, it would not be enough to recover the plaintext of those emails. One would have to cryptanalyze AES to recover a large fraction of those session keys. If that were possible, the security of AES would be called into question for QKD use as well.

Requirement for physical security

A QKD system would cause little damage if stolen (a few recent keys might still be in memory somewhere, though this risk could be minimized with good design). A Terakey stolen intact could cause a major compromise if forward security measures described above were not used or were ineffective. However, in practice both types of system require continuous physical security to prevent tampering. A well-equipped attacker with physical access to either a QKD device or a Terakey system could install modifications that subvert security, say by storing keys and leaking them through side channels. A variety of tools and devices are available to state-level actors, at least, to compromise the

integrity of any device whose continuous physical security is not assured. [18] Strict security is needed, both while hardware is being transported to user sites and on an ongoing basis once installed.

Transportation requirements

Terakeys will have to be prepared at a central location and transported securely to their points of use. In theory, each user could build their own QKD system that conformed to a published protocol, but in practice highly specialized skills, components and tooling would be needed and, in practice, QKD systems will be built in a few factories on the planet. Complete QKD units will require transport to their points of use, which must be secure to avoid tampering on route. Data files to refresh a Terakey will also need secure transport on an occasional basis, but so will spare parts for QKD systems. One advantage for Terakey is that the keys can be broken into two or more components and sent separately.

Speed

Toshiba reported in 2018 a QKD speed in excess of 10 megabits per second over a 7 km fiber [19]. However, speed decreases dramatically with distance, more specifically with channel losses. Korzh, et al, reported a secret key rate of 12.7 kbps at 104 km, dropping to 3.18 bps at 307 km, even with ultra-low-loss optical fibre (51.9 dB loss). [20] A group in China recently achieved QKD using a satellite relay over a distance of 1,120 km with speed of 0.12 bps. They claim a satellite with a more powerful light source could increase speed a hundredfold. [21]

There are two limits on Terakey speed, key memory access times and impact on the fraction of key bytes used (*f*). Several commercial SSD modules offer speeds at the 350K IOPS level for random reads. That speed would allow access at a rate of 2.8 megabits per second. These speeds assume a significant queue depth, but that should be achievable if a Terakey system is operated at high speed. If the Terakey is stored on multiple modules, say ten 1TB units, it should be possible to access the modules in parallel, allowing speeds up to 28 megabits per second. Modern CPUs should be able to handle the associated computational workload. However, keeping *f* low at such high speeds would require a very large Terakey or frequent refreshes. One could assume that an adversary can only recover a small fraction of the symmetric keys generated by the Terakey. That would extend the useful life of the Terakey proportionately.

At speeds in the 10 bps range currently achieved by QKD over longer distances, Terakey system with multi-terabyte memory could operate with low *f* for decades. Less expensive rotating memory could be used to store the Terakey at such low speeds.

Using Terakey along with Quantum Key Distribution

Terakey could be used synergistically with quantum key distribution in situations where fast QKD links are established between a limited number of locations and high-level security is desired for additional locations. An international bank might use QKD between its main offices in different countries, but it might be too expensive for individual branches. QKD could then be used to initialize or update Terakeys for local branch or mobile use within each country.

Conclusion

Terakey is not a universal solution for all communications security problems, but it does address two issues with widely used encryption systems, the lack of mathematical proofs of security, and the risk of surreptitious key exfiltration. It also provides a conventional alternative for comparison with quantum key distribution, offering similar security with less dependence on specialized communication channels, unique manufacturers and exotic, hard-to-audit technology.

Bibliography

[1] V. Scarani, H. Bechmann-Pasquinucci, N. J. Cerf, D. Dusek, N. Lutkenhaus and M. Peev, "The Security of Practical Quantum Key Distribution," *Rev. Mod. Phys./arXiv:0802.4155,* vol. 81, p. 1301, 2009.

[2] U.S. National Security Agency, "Commercial National Security Algorithm Suite," 19 August 2015. [Online]. Available: https://apps.nsa.gov/iaarchive/programs/iad-initiatives/cnsa-suite.cfm. [Accessed 12 April 2020].

[3] C. E. Shannon, "Communication Theory of Secrecy Systems," *Bell System Technical Journal,* vol. 28, no. 4, p. 656–715, 1949.

[4] J. Assange, R. P. Weinmann and S. Dreyfus, "Rubberhose," Wayback Machine, 19 August 2001. [Online]. Available: https://web.archive.org/web/20120716034441/http://marutukku.org/. [Accessed 9 June 2020].

[5] J. Spilman, "Blind Hashing". United States of America Patent 9,021,269, 28 April 2015.

[6] A. Reinhold, "System and method for securely storing and utilizing password validation data". United States of America Patent 10,057,065, 21 August 2018.

[7] E. Mulley-Goodbarne, "Red Pitaya launch latest STEMlab platform," Newelectronics , 27 February 2020. [Online]. Available: https://www.newelectronics.co.uk/electronics-news/red-pitaya-launch-latest-stemlab-platform/224577/. [Accessed 7 April 2020].

[8] Noisecom, "Noisecom products listing," [Online]. Available: https://noisecom.com/products/instruments. [Accessed 7 April 2020].

[9] Maxim Integrated, "Building A Low-Cost White-Noise Generator, Application Note 3469," 14 March 2005. [Online]. Available: https://www.maximintegrated.com/en/design/technical-documents/app-notes/3/3469.html. [Accessed 7 April 2020].

[10] M. S. Turan, E. Barker, J. Kelsey, K. A. McKay, M. L. Baish and M. Boyle, "Recommendation for the Entropy Sources Used for Random Bit Generation, NIST SP800-90B," National Institute of Standards and Technology, Washington, D.C,, 2018.

[11] E. Barker and J. Kelsey, "Recommendation for Random Bit Generator (RBG) Constructions, NIST SP-800-90C (2nd draft)," National Institute of Standards and Technology, Washington, D.C., 2016.

[12] U.S. Army, "Chapter 6, Cryptographic Equipment Destroyer, Incendiary," in *Army Ammunition Data Sheets For Demolition Materials, TM 43-0001-38*, Washington, D.C., Department of the Army, 1981, pp. 6.3, 6.4.

[13] M. Wei, L. M. Grupp, F. Spada and S. Swanson, "Reliably Erasing Data From Flash-Based Solid State Drives," in *FAST '11: 9th USENIX Conference on File and Storage Technologies*, San Jose, CA, 2011.

[14] Apacer, "ATA Secure Erase," Apacer, [Online]. Available: https://industrial.apacer.com/en-ww/Technology/ATA-Secure-Erase. [Accessed 6 April 2020].

[15] R. Kissel, A. Regenscheid, M. Scholl and K. Stine, "Guidelines for Media Sanitization, SP 800-88 Rev. 1," National Institute of Science and Technology, Washington, D.C., 2014.

[16] A. Armstrong, "Samsung 970 EVO Plus 2TB Review," Storage Review, 1 July 2019. [Online]. Available: https://www.storagereview.com/review/samsung-970-evo-plus-2tb-review. [Accessed 7 April 2020].

[17] U.S. National Security Agency, "Media Destruction Guidelines," [Online]. Available: https://www.nsa.gov/resources/everyone/media-destruction/. [Accessed 6 April 2020].

[18] T. Zetter, "NSA Hackers Get the 'Ungettable' With Rich Catalog of Custom Tools," 30 December 2013. [Online]. Available: https://www.wired.com/2013/12/nsa-hacking-catalogue/. [Accessed 12 April 2020].

[19] J. Boyd, "Quantum Technology Promises Practical Cryptography With Unbreakable Keys," *IEEE Spectrum,* 19 Spetember 2018.

[20] B. Korzh, C. C. W. Lim, R. Houlmann, N. Gisin, M. J. Li, D. Nolan, B. Sanguinetti, R. Thew and H. Zbinden, "Provably Secure and Practical Quantum Key Distribution over 307 km of Optical Fibre," *Nature Photonics,* vol. 9, pp. 163-168, 2015.

[21] C. Q. Choi, "Quantum Satellite Links Extend More Than 1,000 Kilometers," *IEEE Spectrum,* 15 June 2020.