Usable everlasting encryption using the pornography infrastructure

Enka Blanchard, Siargey Kachanovich

Usable everlasting encryption using the pornography infrastructure

Enka Blanchard
Digitrust, Loria, Université de Lorraine
Nancy, France
enka.blanchard@gmail.com

Siargey Kachanovich
Corpy&Co., Inc.
Tokyo, Japan

***Abstract*—Nine years before Snapchat and its ephemeral messages, Aumann, Ding, and Rabin introduced the idea of everlasting security: an encryption that could not be decrypted after a certain date, no matter the adversary's computing power. Their method is efficient but not adapted to real-life constraints and cannot effectively be used today.**

This paper looks at what potential entropy sources are available today and proposes a new solution that makes use of the already existing communications from pornography distribution networks. The method proposed has multiple advantages stemming from the fact that pornography is shameful in most societies, giving the agents plausible deniability. It is also usable off-the-shelf by individuals with limited technical skills, although it still requires some effort.

***Index Terms*—Usable security, Random beacon, Bounded memory model, One-time pad**

Between January 5th, 2010 and February 3rd, 2010, Chelsea Manning downloaded more than 491 000 confidential documents, smuggled them through security hidden in a Lady Gaga CD, then onto an SD card hidden in a camera, before sending them to WikiLeaks through Tor [1]. Later in 2016, an anonymous[1] whistle-blower nicknamed "John Doe" managed to communicate 2.6TB of confidential financial documents — known as the Panama Papers — to the International Consortium of Investigative Journalists [2]. In both events, a more attentive supervisor could probably have prevented or at least detected the leak as it was happening. The methods used by Chelsea Manning are most probably by now obsolete, which poses the question of the technological tools available to the whistle-blowers. An issue is that the whistle-blowing attempts of highest importance come from agents within large institutional and corporate actors, which are best suited to detect leaks by storing and decrypting their agents' communications.

Irrespective of the ethical considerations, it then seems natural to look for methods through which an agent within a large organisation could exfiltrate information — ideally to report criminal activities. This agent's main issue then becomes the communication of data, while being watched by an overpowering adversary.

There are encryption methods available today that are currently impossible to decrypt without a key — e.g. any of the current asymmetric encryption standards[2]. For a whistle-blower, however, asymmetrical cryptography cannot be considered entirely secure, as the current methods all depend on computational hardness of certain problems, and this hardness is still just an assumption. An adversary capable of storing the encrypted messages for an arbitrary duration could theoretically either spend the computational resources when they are cheaper, or wait for the development of more efficient decryption algorithms. Furthermore, due to Shannon's theorems, we know that the only way to avoid a leak of information is to use a one-time pad with the same length as the original message [3]. We then require an external source to obtain a one-time pad.

a) Previous work: In an article written in 2002 [4], Aumann, Ding and Rabin introduced the first everlasting encryption algorithm, where an adversary that cannot decrypt the message in a bounded time after its emission will have an exponentially small probability of being able to decrypt it afterwards[3]. The peculiarity of their model is that it holds even against an adversary that obtains unbounded computational capabilities right after the cut-off time. For this, the authors make two central assumptions:

- There exists a common source of public random bits.
- No adversary can store more than a constant fraction of these random bits.

One crucial thing that the authors allow, which makes their model different from a previous model by Maurer [6], is that the adversary has a storage capacity that is limited by a fraction of the number of random bits, but can store data that is different from the random bits themselves. For example, they could store a compressed version of the bits if doable. This seems quite reasonable, as it simply corresponds to the existence of a common source of public random bits that has a very high throughput to avoid being stored. However, implementing it in practice reveals a few challenges. An idea initially mentioned in the original paper was to create a constellation of satellites that beam random bits at a high rate.

[1]The whistle-blower remained anonymous through the usage of Tor and limited communications.

[2]Notwithstanding side-channel attacks, infected client machines and the possibility that an actor could have secretly found an algorithm to polynomially solve supposed hard problems.

[3]This is opposite to timed-release encryption methods, where it becomes feasible after a certain point to decode a message [5] and the adversary is assumed to have infinite computational capabilities.

This is alas costly, and not directly usable by the average user. An alternative proposed by the same authors was to access and compress a large amount of textual web pages, but this creates new problems.

b) Contributions: Our goals in this paper are three-fold. First, we formalise the objections above in an agent model. Second, we propose a list of alternative entropy sources and show that the choice is in fact quite limited. We show that we can use socially generated data as a primitive for cryptography, and that the main candidate seems to be pornography. We also detail how to use it in an appropriate way. Finally, we look at the social implications of using such a system, and why it has advantages beyond the simple cryptographic ones. One central advantage comes from the very taboo surrounding pornography, making this choice a strength of the protocol.

I. Model and intuition

Our goal is to study a model that is adapted to real world use — for example two agents in different organisations wanting to exchange securely. Keeping our introductory example, it could be an agent of a government trying to exfiltrate some files in a whistle-blowing attempt by communicating them to a journalist.

A. Agents

We consider four agents:

- Alice is the agent with the data, who wants to send data to Bob without getting caught or attracting too much attention.
- Bob just wants to receive Alice's data.
- Eve is the supervisor of Alice as well as many other agents, and wants to prevent data leaks. Eve has large capabilities: she can intercept all messages from her agents, and store all the encrypted ones for potential later decryption. However, she cannot store all the information exchanged in clear because of size constraints. Eve also monitors the information exchanged in clear and tries to detect anomalous patterns.
- Carlos represents the rest of the internet.

Here, realistically, Eve could be the head of Alice's organisation (or of its internal security division). She would then be able to impose rules such that most of the communications have to be in clear-text — or easily decrypted — with some allowances for higher security to protect her agents' privacy, at the cost of storing those for future use if she has any hint of wrong-doing. Eve stores every encrypted communication between Alice and Bob, and inspects all the exchanges between Alice and Carlos. However, Eve cannot store all the data exchanged between the internal network and Carlos — representing the rest of the internet — due to space constraints. If she suspects Alice, she might store her communication data, but she cannot store all of it as long as Alice is statistically indistinguishable from her colleagues. Eve possibly has the capability of decrypting some encrypted messages, but this requires time and is expensive.

We could also consider a slightly more complex model where Bob also has a supervisor. Because of symmetries in the protocol, both models are close to equivalent and we focus on the simpler one.

B. Intuition behind the protocol

In such a model, a very simple protocol is available, as imagined in [4]. Using asymmetric encryption — which we can initially assume to be secure for at least a limited time[4] — Alice sends a set of pseudo-random numbers to Bob. Those are pointers to a common source of random bits held by Carlos. After they confirm having the same data — e.g. by the use of a checksum — Alice and Bob can exchange using the equivalent of a one-time pad. As long as Eve does not store the whole data required, she will not be able to decrypt Alice's messages.

The hard task is then to find a good and discreet source of random bits. Alternatively, a public source of low-entropy bits can also work as long as a randomness extractor is used, for example the same extractor used in the original paper [4].

II. Entropy sources

A. Constraints

Before listing potential sources, we have to look at the constraints we're facing.

a) Throughput: The first constraint is quite simple: the data throughput has to be large enough to make storing it be impossible, or at least extremely expensive. In practice, this evolves with the hardware costs, but 1 TB/s is a good initial target to counter all but the largest state actors, as the global cloud storage increases at the rate of 8TB/s [7]. A smaller throughput could still be secure — storing it becoming only extremely expensive — while 10TB/s would be more than enough for the foreseeable future.

b) Canonicity: The second constraint is the canonical nature of the entropy source. As Alice and Bob agree on a set of pointers, these pointers need to target the same data stream, and both agents must obtain the same data when they try to access it.

c) Accessibility: The third constraint is that Alice and Bob should be able to access Carlos' random bits, following Alice's pointers, but in a way that do not set Alice or Bob apart from their colleagues. As such, the data should be common enough, in the sense that it is accessed on a regular basis by a large number of people.

B. Original sources

The original paper investigated two main sources of entropy: satellite-based random beacons, and random web-page compression [4]. Both have some weaknesses, however, which we will mention before looking at potential replacements.

[4]This is reasonable in many cases, unless the main encryption methods are all breakable at a low enough cost for all outgoing messages to be automatically decoded. If that is not the case, Eve would probably have to focus her resources on specific messages and use a queue, in which case it would be natural for Alice to have at least a few hours between the time the message is sent and the time it is deciphered, during which she can get the key and encrypt the data before Eve knows that she should store all communications.

a) Satellites: The first entropy source imagined relies on a satellite — or a small constellation of satellites — that beams random bits. The source being unique, canonicity is evident. An issue is that, as storage cost decreased exponentially since the original proposal, a single satellite is far from being an option. Using SpaceX's Starlink project as an example of satellite constellation, a minimum of 1600 dedicated satellites would be required to achieve the required bound [8], for a total cost above one billion dollars[5]. Depending on whether accessing the satellite system is only used for this purpose, the specialised receiving equipment — which could be costly — could set Alice apart and make her obviously suspect to Eve. Thus, neither throughput nor accessibility is achieved.

b) Random beacons: Although not mentioned in the original paper as it did not exist yet, a simpler possibility would be to use a public random beacon, such as the NIST randomness beacon [9]). This source only produces 512 bits per minute today, but its throughput could be increased. An issue is that, even if it achieved 1TB/s of random bits, the total demand from clients accessing the beacon would probably be much lower than that in practice. As such, Eve could simply store all the random bits requested from the beacon. A coordinated effort could be done to spam the system with bogus requests, but seems unlikely to succeed[6]. Moreover, accessing such a service — or going through an anonymising service such as Tor — would by itself make Alice suspect.

c) Web page compression: A second original method goes by accessing random web pages, compressing them, and using this data as a one-time pad. A naive implementation of this method already fails for simple throughput reasons: considering only pages indexed by Google and ignoring — for now — multimedia content, the total throughput is far from enough. The total index size of Google, for example, is still storable by a powerful adversary [11]. This method also requires both agents to agree on a large set of web pages and to have common access to them (without local variability of content due to redirections, which can be hard to foresee. Accessing these pages might trigger Eve's detection mechanism because of their sheer quantity and potential lack of pattern.

C. Multimedia sources

A different solution from the ones mentioned above is to use the multimedia content present online. Specifically, one can use video sources, as they comprise more than half of the 500TB/s of internet throughput [12]. Non-animated videos also work quite well with randomness extractors as they have a high amount of inherent noise (making lossless compression beyond a ratio 1:2.5 unrealistic [13]). We consider two kinds of traffic: *upstream*, with the communication going from a computer to the network, which is contrasted with *downstream*, which means the communication from the network to computers. Here, we must be careful, as most of the downstream traffic is many-to-one, with the same content being distributed to many clients from a single source. Youtube, for example, represents more than 11% of global downstream traffic and sends more than 50TB/s from its servers to many devices. Despite this, due to the fact that most people seek the same videos, Youtube is still quite storable — and is stored in practice — as its total sizes only increases at a rate of 40GB/s [14]. We are then left with multiple candidates, all in the same category: many-to-many video streaming services with many different sources of content. We will focus mainly on the upstream throughput, as it makes it easier to distinguish high entropy sources by traffic, as multiple sources seldom upload similar streams, when compared to downstream traffic where a single service can have many redundant servers, each doing multicast.

a) Twitch: Twitch — a streaming platform with a focus on video-game streaming — is the first candidate. It has a sufficiently high throughput, with more than 5% of all upstream traffic and 2.2 million active content creators each month [14]. However, using Twitch as a source of random bits is problematic for one central reason, which is that these streams are potentially highly compressible. This is where the choice of attacker model is crucial. It is, in fact, one of the few practical cases where there is a real difference between Maurer's original model and Aumann *et al.*'s modified attacker. In the former, Eve's only choice is to select which bits of the data she keeps because of storage limitations [6]. On the other hand, Aumann *et al.*'s model allows Eve to perform arbitrary computations on the data — e.g. compressing it [4]. As it turns out, Twitch's data is mostly composed of video-game live-stream from a few major video-games. By nature, such streams contain elements that mostly do not change during the stream — such as the user interface of the game. Not only that, the game streams can even be entirely simulated, from such information as the input to the game and the random number generator values. The size of this information is a lot smaller than the entire stream[7], which makes Eve theoretically capable to store entire streams losslessly with extreme compression ratios. Because of this, Twitch fails to achieve sufficiently high throughput for our purposes.

b) Video chat: Our second candidate lies in direct video chats and calls, corresponding to Skype, WhatsApp and competing services. This has a large throughput — at least 8% of upstream internet traffic — and is not easily compressible without high losses. However, it suffers from two problems. First, it is generally not accessible to people outside the call, unless they have advanced surveillance capabilities, making it fail the accessibility constraint. Second, they are distributed, which makes it hard to create any canonical indexing.

c) Pornography: Our last candidate, as strange as it seems, is live pornography. Besides its non-technical advantages detailed in section V, it is the first candidate to truly

[5]Moreover, one could wonder about whether the entities capable of funding such a system have any interest in doing so.

[6]Akin to the famous but fruitless attempt made against the ECHELON system [10].

[7]This adopts the point of view of Kolmogorov complexity, but is also true in practice, as many video-game replay files only weight a few MB per hour, while containing all the information needed to replicate the game.

satisfy all our constraints. It is hard to estimate its throughput accurately, but first order approximations seem to exceed our expectations. At least 4% of Google search requests concern pornography, and the largest live pornography web site (livejasmin.com) is consistently ranked in the top 50 most visited web sites worldwide — behind two other pornography web sites, according to Amazon Alexa[8]. It is nearly impossible to get accurate numbers for the total throughput of live-streaming pornography. We can try, however, to give lower bounds on the example of livejasmin.com, despite it being only one of many alternatives as the market is quite distributed between a large number of actors. Like a few of its competitors, livejasmin.com already has thousands of performers at any point — generally less than ten thousand, however, putting it around one order of magnitude under services like Twitch. These performers often have high definition streams, going from 1MB/s to 3-4MB/s [15]. By itself, this website then has an upstream throughput counted in tens of GB/s, which is already comparable to the storage increase of Youtube. The real upstream throughput can be even higher, as we did not address response streams from viewers. Some of the viewers stream themselves, which can be visible in the main channel. As each channel can have thousands of viewers — of which a fraction might be streamers themselves — this could increase the total throughput by one or two orders of magnitude.

The final reason in favour of live-streaming pornography is that it is quite accessible. This being said, we need to note two caveats. First, there is sometimes a — limited — financial cost to access streams on certain websites. Second, browsing pornography by itself is illegal in many countries — mostly in Asia and Africa. We will come back to this issue later, as for now this source of entropy fulfils most of our requirements. All that is left is to find a way to make it canonical.

III. Protocol to use the live pornography infrastructure

A. *Protocol overview*

We have found a high-throughput, hard-to-compress, distributed and accessible source — which we now denote as Carlos as in our model. Now, Alice and Bob need to agree on the indexing. Here is one potential protocol to address this, with details on the crucial parts shown in the next subsection (while a formal algorithm is shown in the Appendix).

1) Alice sends Bob an initial message using any asymmetric encryption system. This messages contains the parameters for the data held by Carlos, in the form of a set of n complex pointers. Each pointer has data corresponding to a web site[9], the index of a video stream on that web site, a time to start recording the stream, a temporal marker to coordinate the recording, and a duration. The pointers do not correspond to existing videos but to streams in the near future. The time delay depends on the agents' constraints, going from 5 minutes to a few days.
2) Alice and Bob both record the streams pointed to, and extract entropy from those by taking a common start time and using a randomness extractor. This step can be seen as the transmission of data from Carlos to Alice and Bob.
3) Alice and Bob obtain n streams of similar length, and truncate these to equalise the lengths.
4) If Alice only managed to download n' streams instead of n, she still applies the protocol with n' streams instead.
5) Alice computes a parity stream by XORing her n streams and sets it aside to send to Bob later. This will allow Bob to have some redundancy in the case of one stream being not correctly acquired.
6) Alice splits her n streams into blocks of small equal length. She then hashes all the i-th blocks together and concatenates them to obtain a one-time pad.
7) Alice hashes each of her streams independently to get n checksums.
8) Alice sends a message to Bob containing the checksums, the encrypted message and the parity stream she set aside before.
9) As Bob is supposed to have downloaded the same streams, he checks the hashes to ensure that they are indeed equal to Alice's. If Bob is missing one of the streams, he recomputes them using the parity stream.
10) Bob hashes the streams block-wise — exactly as Alice did — to obtain the same one-time pad and decrypt Alice's message.

This protocol has the advantage of requiring only two rounds of one-sided communications from Alice to Bob, with a delay between the two to have the time to record the streams. It also integrates some fault tolerance, to prevent the inevitable transmission errors. The protocol above glosses over the pointers, which we must now investigate.

B. *Making a canonical pointer*

As we stated above, each pointer is composed of five elements: the URL of a web site, the duration of the video to record, the index of a stream on that web site, the approximate time (up to a few seconds) to start recording the stream, and a nonce to coordinate the two recordings. The web site's URL, the time to start recording, and the duration of the video are easy to define. For instance, we can assume for simplicity that each web site is well-defined by its URL [10]. The other two elements — stream index and starting frame — require more work.

a) Stream index: The index of a stream on the web site is harder to agree on, as the stream is not well-defined at the time Alice sends her message. Even worse, streams are generally ordered in a variable way. For example, because

[8]https://www.alexa.com/topsites

[9]A single web site could be used, making this first element obsolete, but it is safer to include it to allow for a diversity of web sites, making it more adaptable and giving the system access to a larger throughput.

[10]URLs can change depending on countries because of redirections, but the underlying streams tend to be the same.

Alice's and Bob's accesses are asynchronous, some streams may disappear in the interval between, hence changing the order of the streams. As such, we must give a pointer that statistically will point towards a single stream, even if both agents do not look it up simultaneously. Finally, the probability of selecting a stream from all streams should be as close as possible to uniform to maintain the throughput guarantees.

One potential solution could be for Alice to select a large number x and send it to Bob along with the pointer. If the web site has k streams when Alice accesses it, she selects the stream number $(x \bmod k)$. Due to the strong variability in the number of streams, this still fails with high probability. We can then pick a large constant c and take the stream number $\left(x \bmod \left\lfloor \frac{k}{c} \right\rfloor\right)$. By taking $\left\lfloor \frac{k}{c} \right\rfloor$ instead of k, we reduce the probability of it changing between the two different accesses.

The number of streams tends to be relatively stable on a given web site, with the previous method being able to absorb most of the minor variability. However, this is still not sufficient for our purpose, as the order between two streams can change much faster than the number of streams. This is especially true when the streams are ordered by number of viewers. A solution is then to choose a second criterion that is independent of the one being used for the sorting. Both Alice and Bob then select the first stream in the list after number $\left(x \bmod \left\lfloor \frac{k}{c} \right\rfloor\right)$ that satisfies this criterion[11].

b) Starting frame: Alice and Bob require the exact same streams for Bob to be able to decrypt the message. This means that they must agree on a starting frame for the video. In practice, they can simply record and download a certain quantity of video, and then manually choose a starting frame. This means that they can easily agree on the start time with a margin of a few seconds, without having to agree on a shared clock beforehand[12].

Along with her pointer, Alice then sends a random value — the nonce — and both she and Bob hash frames from the video until they get a hash whose first bits coincide with the random value. By setting an appropriate precision for this nonce, they can get the same start frame with high probability even if they started recording at slightly different times. However, this means that the delay between the times when Alice and Bob start recording has to be at least one order of magnitude smaller than the duration they expect to record[13]. In practice, assuming the delay in accessing the stream is less than 30 seconds, recording 10 minutes of video is more than sufficient.

With a canonical stream, a set duration and a starting frame, Alice and Bob can both extract entropy from the stream (or losslessly compress them with a good algorithm). By setting $n = 10$ and sending a single parity stream, we already have good guarantees. Eve needs to be lucky and store at least 9 out of the 10 streams. Even if she manages to store 10% of all streams, her probability to decrypt the message is still at most

$$10 \times \frac{9}{10} \times \left(\frac{1}{10}\right)^9 + \left(\frac{1}{10}\right)^{10} \approx 9 \times 10^{-9}.$$

IV. Social aspects

In addition to its cryptographic feasibility, the protocol that we showed has multiple advantages that are not directly technical.

a) Plausible deniability: One aspect that seems to be a huge defect of this protocol is its reliance on pornography, which is badly seen in most societies, and often forbidden (at least in working spaces). It is instead one of the main strengths, for multiple reasons.

First, the very shame associated with pornography is useful as it gives Alice plausible deniability. If she ends up getting caught by Eve, she can try to explain her forbidden behaviour by saying she was looking at pornography, and was trying to hide her shameful behaviour. Humans do not always perform well when they are interrogated, and having a — less serious — crime to confess to is a boon. Although it might be a stretch, she could also try to explain the access to potentially restricted files she was trying to exfiltrate as being linked to a virus downloaded at the same time as the pornography.

One issue with the protocol is that downloading pornography can trigger some alarms on certain networks. This is not necessarily a bad thing: getting caught downloading pornography would probably lead to a disciplinary hearing, which would confirm Alice's suspicions that her communications are scrutinised, without compromising her as much as directly trying to access cryptographic resources. Moreover, downloading pornography wouldn't make Alice a statistical anomaly, quite the opposite: 59% of respondents to an online study admitted to having accessed pornography web sites from their office [16], and many polls reveal similar behaviours (alas with low data quality). Firing all the people caught doing so because of security risks would be costly to Eve because of the sheer number of false positives. Any suspicious behaviour of Alice — such as hiding a USB key on which the pornography streams are recorded — then becomes partially justified without incriminating Alice further than for the lighter offence of watching pornography at work.

It is true that the applications of this line of reasoning depend on where Alice works. For example, if she works in a critical infrastructure like a nuclear plant, the pornography data might make her visible to Eve. However, it would probably not be the case if Alice works for a bank or on an army base.

b) Immediate employability: The second advantage of the protocol is that it can be used directly, without advanced tools or the creation of a large source of entropy. With the values shown previously, anyone could send a secure email to their interlocutor, with a list of web sites, stream indices, different times to access them, duration and nonces. As the system can tolerate a 30 second difference, it is doable manually without requiring automation of any task. The only

[11]If we are looking at response streams, we can use this method recursively.

[12]This would be made even harder by the fact that Alice and Bob can have different latencies.

[13]A similar method could be used for the end time, but getting a duration is safer and increases the chance of all n streams having broadly similar bit-length.

step left is agreeing on a starting frame and computing the one-time pad, which can be done using off-the-shelf tools.

c) Public reaction to surveillance: Finally, although government surveillance of citizens' online activity is now partially tolerated, the public is much more critical of it when it comes to intimate subjects. The public outcry after Edward Snowden's revelations that government agencies stored and accessed sensitive personal data is an example of this [17]. This strongly negative image would make it harder for most countries' agencies to receive the massive funding required to store even a portion of the streams considered[14], especially in Europe after the implementation of new directives on the right to be forgotten.

V. Issues and extensions

a) Encoding variability: One technical issue with the protocol we proposed is that the video stream can be different due to variations in the stream's encoding. There are multiple ways to address this, depending on Alice and Bob's respective constraints. The first is to hash the received frames (by groups of a few dozen frames), exchange the hashes, and only keep the common ones. However, this requires an additional intermediate round of communication going from Bob to Alice. An alternative is to use locality-sensitive hashing (LSH) [20] to ensure that the stream is similar on both ends. This strongly reduces entropy, but maintains the 2-round one-sided communication structure.

b) Improving stream agreement: With the proposed protocol, if the number of streams k changes between the different access times, the probability of disagreement is $\frac{1}{c}$. This can be further improved at a small cost to Alice and Bob. Instead of checking k upon getting to the web site, they instead refresh the counter and track its evolution for a minute. Many methods then become available to Alice and Bob, for example, they can simply keep track of the maximum and minimum reached by k, and then use stream $\left(x \bmod \left\lfloor \frac{k_{\max}+k_{\min}}{2c} \right\rfloor\right)$. This increases their agreement probability to at least $\gamma^2 \times \frac{c-1}{c}$, where γ is the proportion of overlap on the time they spent looking at the evolution of k.

c) Increasing the fault tolerance: The protocol can only correct one missing stream in its current state. It can potentially be extended to tolerate more erroneous or missing streams. Thanks to the checksums sent by Alice, Bob can eliminate the erroneous streams — or even try to fix them, which can be done with probability $\frac{1}{2}$ if Bob's starting frame is different from Alice's, but he recorded a bit more than the expected duration. We can then reduce the problem to that of fixing missing streams instead of erroneous ones. To go further, one could use double error correction as in RAID 6 and derivative systems [21]. It would also be possible to use a method based on Shamir's secret sharing to create some simple redundancy [22], by sending some additional data with Alice's second round message.

The question to ask is therefore: how many missing or erroneous streams should be tolerated, compared to the number of total streams? This proportion should be quite higher than the proportion of public data that Eve is supposed to be able to store, as otherwise it increases her chance of decrypting the message. However, Alice can set a high fault tolerance ratio if she has a single opportunity to send the message and wants to be sure that it gets decrypted. This would still be secure if it is a rare occurrence, as Eve would not have the incentives to invest into the storage necessary to decrypt just a few messages with strong redundancy.

d) Addressing already compromised encryption methods: If we decide to push Aumann *et al.*'s fears of an all-powerful adversary further, we can get a slightly different model that is in practice quite realistic. It is imaginable that Eve could already have functioning attack schemes against certain encryption methods. This is consistent with recent events, such as what happened with the NIST SP800-90 Dual Elliptic curve PRNG [23]. In such a case, Alice can send two or three pointers, each in its own message encrypted with a different encryption algorithm. She should, however, be careful with the parity checks to prevent the decryption of one source from revealing the whole secret.

VI. Discussion

The main contribution of this paper is a protocol to exfiltrate secrets that can realistically be used today by people with limited technical skills. It is a concern that our protocol can be used by actors with nefarious intents as well as well-meaning whistle-blowers. As such, it is not politically neutral. We believe, however, that it would have limited effect on governmental and industrial espionage. Indeed, those activities generally benefit from advanced technical expertise and highly-refined toolkits due to the powerful financial interests involved. On the other hand, whistle-blowers generally do not benefit from such a support network and would be the primary users of this technology.

One important limitation of this protocol is that it relies on Alice and Bob having access to live pornography, which presupposes two things. First, they must both have reliable internet access. Second, they must also have legal access to pornographic websites, or at least access without being an anomaly. For example, citizens of most European countries fulfil both conditions, whereas people in China would not be able to use this protocol at all[15] [25]. In states like Indonesia, the situation is mixed, as despite the pornography being illegal [26], there currently exists a wide pornography consumption in the country [27]. That said, the lax enforcement of the

[14]Of course, most intelligence budgets are not made entirely public. Considering prices similar to that of the cheapest commercial alternatives — such as Amazon Glacier [18] — storing data costs close to 0.005$/GB per year. Considering a hypothetical stream at 100GB/s, the costs over the first year would be more than $70 million, and the yearly costs would increase to reach $1 billion over less than a decade. This makes it unaffordable for all but the USA intelligence agencies, as very few countries have intelligence budgets counted in billions of dollars, and none besides the USA have budgets much larger than $5 billion [19].

[15]As Chinese traffic is quite different on many criteria, including the propensity to use certain multimedia content such as audio versus text messages, alternatives might still be possible [24].

law and the fact that pornography is still widely accessed would mostly allow the police to arbitrarily target certain users. It is true that Alice and Bob might invite additional scrutiny by looking at pornography from work, but two factors limit this. First, this behaviour is far more widespread than most people believe, as mentioned earlier. Second, the amount of pornography needed is in fact pretty limited. Assuming wide margins, encrypting 100MB of data would require about 150MB of pornographic videos, which could be obtained in a few minutes[16]. For countries where the access to pornography is limited, there are alternatives that satisfy the constraints presented in this paper and could be used instead of our protocol. For example, we could use slightly altered data from P2P networks. We could also create a large entropy source by introducing some noise into the content distribution system of a P2P network. A similar idea could work by slightly modifying the Scuttlebutt protocol [28]. Contrary to the system we propose, these alternatives are not directly employable today, as they require the cooperation of a large set of users. Moreover, they would require a higher technical ability to implement.

Other entropy source might also appear as the result of two things: the evolution of our online behaviour and the increasing traffic from the deployment of the Internet of Things (IoT). As it stands today, the main sources of upstream traffic that account for more than a few percent either have widely repeated data (such as bittorrent, which can account for up to 30% of upstream traffic), or they tend to be hard to access (like VoIP) and decentralised (like social media video uploads). Twitch is the main contender, but the average compressibility of its feeds should be evaluated before it is used. On the positive side, the recent tendencies in the relative hardware costs for data storage compared to global throughput are currently working in our advantage. If these trends continue, sources that today represent a smaller percentage of global throughput than pornography could become sufficient for our protocol in the future.

Social behaviours online have garnered a lot of interest, especially when it comes to security [29], [30] or to subjects that can be taboo like pornography [31]. Besides just analysing these behaviours, we were interested in how we could build security features based on the social behaviours without directly affecting them. This is but one example and there might be many more social effects awaiting to be used as primitives to improve our security and privacy online.

[16]If the data to be encrypted is much larger, one could potentially relax the security requirements of using a one-time pad, and re-use it a few times by splitting the data, at the risk of letting Eve probabilistically deciphering some fraction of the data.

References

[1] C. Madar, *The passion of Bradley Manning: The story behind the Wikileaks whistleblower*. Verso Books, 2013.

[2] F. Obermaier, B. Obermayer, V. Wormer, and W. Jaschensky, "About the panama papers," *Süddeutsche zeitung*, 2016.

[3] C. E. Shannon, "Communication theory of secrecy systems," *Bell system technical journal*, vol. 28, no. 4, pp. 656–715, 1949.

[4] Y. Aumann, Y. Z. Ding, and M. O. Rabin, "Everlasting security in the bounded storage model," *IEEE Transactions on Information Theory*, vol. 48, no. 6, pp. 1668–1680, 2002.

[5] R. L. Rivest, A. Shamir, and D. A. Wagner, "Time-lock puzzles and timed-release crypto," USA, Tech. Rep., 1996.

[6] U. M. Maurer, "Conditionally-perfect secrecy and a provably-secure randomized cipher," *Journal of Cryptology*, vol. 5, no. 1, pp. 53–66, 1992.

[7] Cisco, "Global cloud index: Forecast and methodology, 2016-2021," Cisco, Tech. Rep., 2018. [Online]. Available: https://web.archive.org/web/20190320151956/https://www.cisco.com/c/en/us/solutions/collateral/service-provider/global-cloud-index-gci/white-paper-c11-738085.html

[8] I. del Portillo, B. G. Cameron, and E. F. Crawley, "A technical comparison of three low earth orbit satellite constellation systems to provide global broadband," in *69th International Astronautical Congress 2018*, 2018.

[9] M. J. Fischer, M. Iorga, and R. Peralta, "A public randomness service," in *Proceedings of the International Conference on Security and Cryptography – SECRYPT*. IEEE, 2011, pp. 434–438.

[10] P. Dykstra. (2001) Net activists launch campaign to jam 'echelon'. [Online]. Available: https://web.archive.org/web/20040109233032/http://edition.cnn.com/2001/TECH/internet/07/30/echelon.protest/index.html

[11] A. van den Bosch, T. Bogers, and M. de Kunder, "Estimating search engine index size variability: a 9-year longitudinal study," *Scientometrics*, vol. 107, no. 2, pp. 839–856, 2016.

[12] Cisco, "Global visual networking index: Forecast and trends, 2017-2022," Cisco, Tech. Rep., 2018. [Online]. Available: https://web.archive.org/web/20190323025839/https://www.cisco.com/c/en/us/solutions/collateral/service-provider/visual-networking-index-vni/white-paper-c11-741490.html

[13] K. Kawaharada, K. Ohzeki, and U. Speidel, "Information and entropy measurements on video sequences," in *2005 5th International Conference on Information Communications Signal Processing*, 2005, pp. 1150–1154.

[14] C. Cullen, "Global internet phenomena report," Sandvine, Tech. Rep., 2018.

[15] L. Pressly, "Cam-girls: Inside the romanian sexcam industry," *BBC News, Bucharest*, 2017.

[16] T. McDonald. (2018) How many people watch porn at work will shock you. [Online]. Available: https://web.archive.org/web/20180205170953/https://sugarcookie.com/2018/01/watch-porn-at-work/

[17] K. Hill. (2014) NSA responds to Snowden claim that intercepted nude pics 'routinely' passed around by employees. [Online]. Available: https://web.archive.org/web/20140720001012/https://www.forbes.com/sites/kashmirhill/2014/07/17/nsa-responds-to-snowden-claim-that-intercepted-nude-pics-routinely-passed-around-by-employees/

[18] "Amazon S3 Glacier pricing (Glacier API only)," 2019, accessed: 2019-09-03. [Online]. Available: https://aws.amazon.com/glacier/pricing/

[19] C. Hippner, "A study into the size of the world's intelligence industry," Master's thesis, Mercyhurst College, 2009.

[20] M. Slaney and M. Casey, "Locality-sensitive hashing for finding nearest neighbors [lecture notes]," *IEEE Signal processing magazine*, vol. 25, no. 2, pp. 128–131, 2008.

[21] C. Jin, H. Jiang, D. Feng, and L. Tian, "P-code: A new raid-6 code with optimal properties," in *Proceedings of the 23rd international conference on Supercomputing*. ACM, 2009, pp. 360–369.

[22] A. Shamir, "How to share a secret," *Communications of the ACM*, vol. 22, no. 11, pp. 612–613, 1979.

[23] T. C. Hales, "The NSA back door to NIST," *Notices of the AMS*, vol. 61, no. 2, pp. 190–19, 2013.

[24] J. Horwitz, "Stop texting right now and learn from the Chinese: there's a better way to message," 2015. [Online]. Available: https://qz.com/443441/stop-texting-right-now-and-learn-from-the-chinese-theres-a-better-way-to-message/

[25] Z. Huang, "Watching porn on China's censored internet is an infinitely evolving cat-and-mouse game," 2017. [Online]. Available: https://qz.com/1001366/how-the-chinese-watch-porn-on-chinas-censored-internet/

[26] M. Lim, "The Internet and Everyday Life in Indonesia: A New Moral Panic?" *Bijdragen tot de taal-, land- en volkenkunde / Journal of the Humanities and Social Sciences of Southeast Asia*, vol. 169, no. 1, pp. 133 – 147, 2013. [Online]. Available: https://brill.com/view/journals/bki/169/1/article-p133_8.xml

[27] G. M. Hald and T. W. Mulya, “Pornography consumption and non-marital sexual behaviour in a sample of young Indonesian university students,” *Culture, Health & Sexuality*, vol. 15, no. 8, pp. 981–996, 2013, pMID: 23782270. [Online]. Available: https://doi.org/10.1080/13691058.2013.802013

[28] R. van Renesse, D. Dumitriu, V. Gough, and C. Thomas, “Efficient reconciliation and flow control for anti-entropy protocols,” in *Proceedings of the 2nd Workshop on Large-Scale Distributed Systems and Middleware*, ser. LADIS ’08. New York, NY, USA: ACM, 2008, pp. 1–7.

[29] M. L. Mazurek, S. Komanduri, T. Vidas, L. Bauer, N. Christin, L. F. Cranor, P. G. Kelley, R. Shay, and B. Ur, “Measuring password guessability for an entire university,” in *Proceedings of the 2013 ACM SIGSAC Conference on Computer Communications Security*, ser. CCS ’13. New York, NY, USA: ACM, 2013, pp. 173–186.

[30] A. Das, J. Bonneau, M. Caesar, N. Borisov, and X. Wang, “The tangled web of password reuse.” in *NDSS*, vol. 14, 2014, pp. 23–26.

[31] A. Mazières, M. Trachman, J.-P. Cointet, B. Coulmont, and C. Prieur, “Deep tags: toward a quantitative analysis of online pornography,” *Porn Studies*, vol. 1, no. 1-2, pp. 80–95, 2014.

VII. Appendix

The following algorithms show the normal version of the protocol with a limited amount of error correction (assuming that the streams are recorded in a similar fashion, otherwise the extensions of Section IV have to be used).

The first algorithm is used by Alice to send Bob the pointers. They then both execute the recording algorithm. Alice then launches her second round algorithm and sends the result to Bob, who obtains the original data through his decrypting algorithm.

```
Data: Size of secret data $L_{DATA}$, desired number of streams $n$
$L := \lceil L_{DATA}/256 \rceil \times 256$
/* Padding the length for simplification */
$nonce = 2 \times \log_2 L$
/* Creating a nonce of appropriate size to end up
   agreeing on the same start frame. It should also
   have a minimum required size if the secret data is
   short. */
$c := 50$
/* c should be higher than the average variation in
   number of streams) */
for $i$ from 1 to $n$ do
    $Pointer[i].URL$ := random pornographic website url
    $Pointer[i].index := Random(100000)$
    $Pointer[i].nonce := nonce$
    $Pointer[i].quotient := c$
    $Pointer[i].duration := L$
    $Pointer[i].time$ := reasonable time/date for Bob to start recording
end
Send Bob $(n, Pointer[])$ using any asymmetric cryptography method.
```

Algorithm 1: Alice first round algorithm

```
Data: number of streams $n$, Pointer list $Pointer[]$
for $i$ from 1 to $n$ do
    Go to $Pointer[i].URL$
    At time $Pointer[i].time$, refresh the page
    $k$ := total number of streams
    $\alpha := Pointer[i].index$ mod $\lfloor \frac{k}{Pointer[i].quotient} \rfloor$
    $Stream[i]$ := record stream number $\alpha$ for $Pointer[i].duration$
end
```

Algorithm 2: Common recording algorithm

```
Data: Secret data $DATA$, number of streams $n$, Streams $Stream[]$,
      Pointer list $Pointer[]$, stream length $L$
begin
    $Good\_str := []$
    for $i$ from 1 to $n$ do
        $Trunc\_str[i]$ :=TRUNCATE($Stream[i], Pointer[i].start, L$)

        /* Truncating each stream to the same number
           of bits by looking for the first hashed
           segment of 256 bits that is compatible with
           the start nonce. */
        if LENGTH($Trunc\_str[i]$) $= L$ then
            PUSH($Good\_str[], Trunc\_str[i]$)
            /* Removing the streams with insufficient
               information (or the ones which failed to
               record). */
    end
    $m$ :=LENGTH($Good\_str[]$)
    $Parity\_str := 0$
    for $i$ from 1 to $m$ do
        $Parity\_str$ :=XOR($Parity\_str, Good\_str[i]$)
        $Check\_sum[i]$ :=SHA1($Good\_str[i]$)
    end
    $One\_Time\_Pad := []$
    for $j$ from 0 to $L/256 - 1$ do
        $Block[j] := []$
        for $i$ from 1 to $m$ do
            PUSH($Block[j], Good\_stream[i][256 \times j]$, 256 bits)
        end
        PUSH($One\_Time\_Pad$, SHA-256($Block[j]$))
    end
    $Encrypted\_Data$ :=XOR($DATA, One\_Time\_Pad$)
    SEND($m, Check\_sum[], Parity\_str, Encrypted\_Data$)
end
```

Algorithm 3: Alice second round algorithm

```
Data: Initial number of streams $n$, Pointer list $Pointer[]$, Received number
      of streams $m$, List of checksums $Check\_sum[]$, Recorded streams
      $Stream[]$, Parity stream $Parity\_str$, stream length $L$
begin
    $Good\_str := []$
    $Final\_str := []$
    for $i$ from 1 to $n$ do
        $Trunc\_str[i]$ :=TRUNCATE($Stream[i], Pointer[i].start, L$)

        if LENGTH($Trunc\_str[i]$) $= L$ then
            PUSH($Good\_str[], Trunc\_str[i]$)
    end
    $k_1$ :=LENGTH($Good\_str[]$)
    for $i$ from 1 to $k$ do
        if SHA1($Good\_str[i]$) $\in Check\_sum[]$ then
            PUSH($Final\_str[], Good\_str[i]$)
    end
    $k_2$ :=LENGTH($Final\_str[]$)
    if $k_2 < m$ then
        if $k_2 < m - 1$ then
            return Failure
        else
            $Final\_str[m] := Parity\_str$
            for $i$ from 1 to $m - 1$ do
                $Final\_str[m]$:=XOR($Final\_str[m], Final\_str[i]$)

            end
            /* This section seeks to correct one
               missing stream, or returns failure if
               more than one common stream is missing.
               */
        end
    $One\_Time\_Pad := []$
    for $j$ from 0 to $L/256 - 1$ do
        $Block[j] := []$
        for $i$ from 1 to $m$ do
            PUSH($Block[j], Final\_stream[i][256 \times j]$, 256 bits)
        end
        PUSH($One\_Time\_Pad$, SHA-256($Block[j]$))
    end
    $Decrypted\_Data$ :=XOR($Encrypted\_Data, One\_Time\_Pad$)
    return $Decrypted\_Data$
end
```

Algorithm 4: Bob decrypting algorithm.