Encrypt to forget

Constantinos Patsakis

Universitat Rovira i Virgili Department of Computer Engineering and Maths, UNESCO Chair in Data Privacy, Av. Països Catalans 26 43007 Tarragona, Catalonia.

***Abstract*—A recent proposal in European Union has raised several talks regarding the right of someone to be forgotten. In this work we are not dealing with the ethical issues that this proposal poses, however we discuss an approach were data are not deleted, but gradually distorted. Data are eroded as time goes by, the constant erosion allows for several time the data to be completely accessible, yet after parsing the data many times, the data decryption becomes more and more difficult. After a certain error threshold, the data become unusable, therefore digital oblivion is achieved not by record deletion, but through making them nonfunctional.**

***Index Terms*—error correcting codes, encryption, right to be forgotten, Privacy, Digital oblivion, Digital forgetting**

I. Introduction

Quite recently several talks have been sparkled after a proposal by V. Reding [1], a European Union Justice Commissioner about the right of someone to be forgotten. The proposal is quite controversial in regard to several other EU directives, for example Data Retention and Data Protection Directives. Surely this proposal triggers many ethical issues, which are of course beyond the scope of this work. This work tries to approach the implementation of such process with another more "human" way. The main scope of this work is to illustrate that we already have the mechanisms to apply closer to real life approaches than "delete row", "delete file", where the content simply vanishes into thin air.

In our daily life we keep forgetting things as more memories are "pushing" their way through. We mix dates, memories, as some extra "noise" is being added. When the noise is small we are able to remember, with much details the facts, yet as time goes by, the added "noise" is fading our memories, discarding some details. So the memories are not completely forgotten many times, they are distorted. A brief categorization of the reasons why we forget as given in [3] is the following:

- Storage failure. The term might seem too technical, yet in many cases, specially when it comes to physical problems, we are not able to store information properly in our brain. The problems stem from many reasons, like problematic encoding, lack of proper memory span of short-term memory, or even lack of elements and enzymes.
- The decay of memory traces. According to this theory, every time new information has to be stored a new memory trace is being created. According to this theory, these memory traces are fading and disappear with the pass of time, if information is not retrieved and rehearsed.
- According to interference theory some memories compete and interfere with each other, therefore when information is very similar to a previously stored, interference is very likely to occur.
- Motivated forgetting is another approach. In this case we try and manage to forget memories, for example traumatic or disturbing events and experiences.

Of course forgetting apart from putting us in awkward situations, is very beneficial for out daily lives. As mentioned above, we forget things that have heart us, from physical pain to discomfort and painful sentiments, enabling ourselves to continue with our lives and in many cases forgive.

The general rule that we have in our implementations to make machines forget is to erase content. This is done mainly by either releasing the locks of the file system on the content, so that other data can be written on the same area of the storage medium, either wiping the space by writing dummy data on top.

In [4], the author regards that the use of expiration dates could prove to be an adequate solution towards enforcing digital forgetting. The idea is that information should be stored in digital storage, associated with an expiration date, after which the information is removed. Moving forward, the author goes believes that future digital storage devices will be able to automatically delete information on the expiration date.

Beyond expiration dates, the core idea of this work is to create the necessary infrastructure on a server that allows it to partially corrupt the data with the pass of time, or their usage. The desired corruption is obviously more closer to the human notion of forgetting, therefore it is a novel approach on the topic. The proposed method tries to keep user's data beyond the server management, the server has not direct access to the original stored information, but to an encrypted version of them. The only prerequisite that we have in our approach is that we need the server to act fairly, meaning that the server will not keep previous copies of the data and will always follow the proposed scheme.

The next section provides some background information about previous work, closely related to this research. Afterwards, we discuss the Hemenway and Ostrovsky encryption scheme which will be the core utility for the proposed scheme. Then the proposed protocol for digital oblivion is illustrated, followed by two sections, one on the implementation issues and one regarding its possible applications. Finally we finish with some conclusions and ideas for future work.

II. Previous work

In order to be able to encrypt and decrypt messages, the applied functions have to be invertible, so in most cases we have bijections. Typical examples are most block ciphers, if

we take AES we have a mapping of 64 bits strings to 64 bits strings, taking as input the message and the key. So m becomes $c = E_k(m)$. If we alter just one bit of the output, so we have c' and try to decrypt it with the same key, then we will end up with $D_k(c') = D_k(c \oplus \overrightarrow{e}) = m'$, where $\overrightarrow{e}$ is the added error vector. Obviously, $m \neq m'$, and due to AES non-linear natute, m and m' will have many differences, hence error correction will not be possible with such algorithms.

One could argue towards an encrypt-then-encode scheme. The scheme is quite efficient, as the only extra cost can be regarded as the extra storage that an error correcting encoding would demand and the processing cost of decoding on the client side. Its high efficiency leaves a backdoor for possible attacks. Since the added noise is after the encryption, the noise is not embedded in the message. Therefore, the data on the server can always be"refreshed", meaning that they can be restored to their original form, by performing error correction to all distorted fields of the database.

A special form of encryption algorithms like [17], [18] which are based on private key encryption schemes have already been proposed, yet several attacks [19], [20] are questioning the security that they may provide.

In the case of public key cryptography, homomorphic encryption could provide a solution to our problem, yet the implementation cost is quite big. Since we want to add some noise to our data the selection of a typical public key algorithm like RSA, which preserves the multiplication, does not fit the purposes. For adding noise we need to apply the XOR on the messages, which points out the Goldwasser–Micali algorithm [22]. Again its implementation cannot be considered efficient, since we need an RSA-long message for encrypting each bit of the message, moreover the algorithm does not have the necessary error-correction features.

The well known McEllice algorithm [10] has error correcting features, after all its security is based on decoding error correcting codes, yet the needed key size and the amount of errors that can be corrected without making any compromises on the security of the algorithm exceed the needs of the scheme that we propose. Based on variations of McEllice algorithm are the aforementioned Rao-Nam scheme as well as Sun's [21], yet they are used for private key encryption.

Other proposed schemes in the literature include [8], [9]. The lack of scalability in Kak's scheme made it not useful, even though is based on another idea, that of D-sequences, decimal expansions of fractions.

III. The Hemenway and Ostrovsky scheme

In [5] Ostrovsky, Pandey and Sahai introduced a construction of a constant information-rate, constant error-rate locally-decodable code, using a private key encryption algorithm. Yet in our proposal we are going to use its public key variation which was later introduced in [6], [7] by Hemenway and Ostrovsky.

Their idea is based on a variation of the $\Phi-$hiding assumption, originally proposed by Cachin, Micali and Stadler [15], requiring only the security of by Gentry-Ramzan PIR scheme [16]. If a prime p divides $\phi(n)$, we say that n Φ-hides p.

Definition 1. Let $\mathcal{P}_k$ the set of primes of bit-length $\frac{k}{2}$, $\mathcal{H}_k$ be the set of products of two primes in $\mathcal{P}_k$ and let $\mathcal{H}_k^{\pi} \subset \mathcal{H}_k$ denote the set of composite moduli that $\Phi-$hide π, i.e.

$$\mathcal{H}_k^{\pi} = \{m : m = pq, \{p, q\} \subset \mathcal{P}_k,\ p \equiv 1\ mod\ \pi\}$$

Small Primes $\Phi-$Hiding Assumption. For all small prime powers, $\pi_0,\ \pi_1$ such that $3 < \pi_0 < \pi_1 < 2^{\frac{k}{4}-1}$, given $b \in_R \{0, 1\}$ and $m \in \mathcal{H}_k^{\pi_b}$, for all probabilistic polynomial-time algorithms A, we have:

$$Pr\left[A(\pi_0, \pi_1, m) = b\right] \leq \frac{1}{2} + \nu(k),$$

for some negligible function $\nu(k)$, where the probability is taken over all $m \in \mathcal{H}_k^{\pi_b}, b \in \{0, 1\}$ and the internal randomness of A.

To clarify the above assumption, we assume that if we are given n a RSA modulus and two small prime numbers p_1 and p_2 from which only one divides $\phi(n)$, then there is no polynomial time algorithm to determine which one of them is the divisor, with probability more than 50%.

So according to Hemenway and Ostrovsky we have:

A brief outline of their encryption scheme is the following. To generate the public key we start by picking t distinct prime numbers $p_1, \ldots, p_t$ such that $5 \leq p_1 < p_2 < \ldots < p_t$. We set $c_i = \left\lfloor \frac{k}{4 \log p_i} \right\rfloor$, so c_i is the largest integer for which $\log p_i^{c_i} < dk$, where $d < \frac{1}{4}$ and we set $\pi_i = p_i^{c_i}$, where $\log$ is computed in base 2 to show the bit length. We generate a random permutation σ of the symmetric permutation on t elements S_t. We then generate the $m_1, \ldots, m_t$ moduli such that $m_i \in \mathcal{H}_k^{\pi_{\sigma(i)}}$. If g_i are the generators of G_{m_i} then the public key is the t-tuple:

$$((g_1, m_1, \pi_1), \ldots, (g_t, m_t, \pi_t))$$

and the private key the $(t+1)$-tuple:

$$\left(\sigma, \frac{\phi(m_1)}{\pi_{\sigma(1)}}, \ldots, \frac{\phi(m_t)}{\pi_{\sigma(t)}}\right)$$

To encrypt a message X we break it in $\frac{n}{lk}$ blocks X_i of size lk. For each block we compute:

$$\widetilde{X}_i = X_i\ mod\ \pi_{(i-1)s+1} \ldots \pi_{is}$$

We pick a random number $r \in \{0, \ldots, \pi_1 \ldots \pi_t\}$ and calculate:

$$h_i \equiv g_i^{\widetilde{X}_i + r\pi_1 \ldots \pi_t} mod\ m_i$$

We then apply the binary Error Correcting Code ECC to each block so the encryption of message X is the t-tuple:

$$(ECC(h_1), ECC(h_2), \ldots, ECC(h_t))$$

For the decryption process we firstly decode each $ECC(h_j)$ to get $c_j = h_{\sigma^{-1}((i-1)s+1)}$. We then decrypt each c_i by calculating:

$$h_{\sigma^{-1}((i-1)s+1)}^{\phi(m_i)/\pi_i} mod\ m_i,$$

to obtain $a_j \equiv X_i\ mod\ \pi_{(i-1)s_j}$. Finally we use Chinese Remainder Code Decoding Algorithm [14] to reconstruct each X_i from $a_1, \ldots, a_s$.

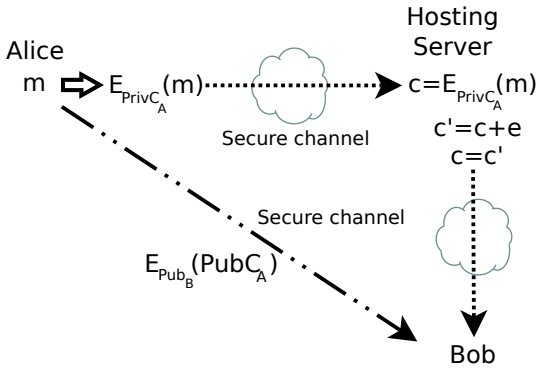

Figure 1. The proposed method.

IV. Proposed solution

As inferred from the previous section, the idea is that in order to enable digital oblivion, we will add noise to the message. This technique will corrupt the information little by little, so the digital oblivion is made gradually until the information becomes unrecoverable. The proposed solution simulates what happens with RAM when the computer is shut down. Since there is no electricity the circuits lose their charge, so the information that is stored starts to corrupt, so depending on the materials in few minutes the content becomes unaccessible.

Every time a file is parsed noise is being added, so after some time the noise passes a threshold and the error correction cannot be made, resulting in several parts of the information to be forgotten. After some time the data is completely useless since their decryption ends up to random data hence the information completely forgotten.

Having the aforementioned scheme of Hemenway and Ostrovsky for encryption, we will now discuss how this can be used to enable people to preserve their right to be forgotten. In order to make the proposal as clear as possible, we will try to illustrate it through an example scenario. The proposed scheme consists of three entities Alice, Bob and Carol. Alice and Bob are registered to Carol's hosting service. As mentioned previously Carol will act fairly, meaning that she will not keep any previous versions of the shared content and the provided keys will be the correct ones. In order for Alice, Bob and Carol to exchange their messages, a secure channel is being used and is going to be taken for granted.

In order to protect user data from eavesdropping from the server side, the content is stored encrypted, unless the data is considered public.

When Alice wants to share message m on Carol's server, she creates a private-public key pair using Hemenway and Ostrovsky algorithm, $PrivC_A$ and $PubC_A$ respectively. This pair will be kept secret from Carol and will be used only when Alice wants to share a message that can be corrupted. Alice will then encrypt the file using her private key $E_{PrivC_A}(m)$ and send it through a secure channel to Carol. If she wants Bob to access m, Alice obtains Bob's public key from Carol and sends message $c = E_{Pub_B}(PubC_A)$ to Bob. Bob can now decrypt c using his private key and obtain $PubC_A$ to decrypt m. Every time that Bob accesses the content, Carol distorts the content and updates it. So every time Carol sends Bob c_i, where $c_0 = E_{Pub_B}(PubC_A)$, $c_1 = c_0 \oplus e_1, \ldots$, $c_i = c_{i-1} \oplus e_i$ and e_i is a random error vector. On receiving c_i Bob applies the decryption algorithm from the previous section and tries to resolve m. If the total error vector $e = e_1 \oplus e_2 \oplus \ldots \oplus e_i$ exceeds the threshold that Alice has set with the initialization of the algorithm, m cannot be recovered. The proposed scheme can be seen in figure 1.

Obviously, the whole scheme depends on having a fair server, meaning that the server will always apply the protocol, which is the only prerequisite of the scheme. If the server does not apply the random noise or keeps previous versions of the shared content, the proposed scheme simply cannot work. A P2P extension of the scheme could of course circumvent the fairness restriction of the scheme.

It is apparent that in the proposed scheme the Ostrovsky, Pandey and Sahai could have been used. The use of a public key algorithms generally enables more application features and ensures Bob about the origin of the content.

Compared to the Goldwasser–Micali encryption scheme, which enables XORing through its homomorphic properties, the Hemenway and Ostrovsky scheme not only enables us to embed more data but simultaneously supports error-correction. The error-correction capabilities of the Hemenway and Ostrovsky encryption algorithm outperform those of McEllice algorithm. Moreover the use of Hemenway and Ostrovsky encryption scheme enables us to use locally decodable codes, which work better than usual error-correcting codes when processing "big" messages.

Since the proposed scheme is designed for sharing arbitrary messages and current trends towards file sharing are big files, applying intentionally random error vectors might result in completely corrupted blocks of data. Therefore, locally decodable codes may prove to be more efficient to decoding, both in time and the efficiency of error correction, compared to common error correcting codes. For more on locally decodable codes the reader may refer to [14].

V. Implementation issues

The storage cost of the proposed solution is $\frac{\rho c}{d}$ times the size of the original message, where

- ρ is the expansion factor of the CRT ECC,
- c is the expansion factor of the error correcting code ECC and
- d is the fraction of bits that can be Φ-hidden.

Therefore, depending on how much error correction we need for our implementation, parameters ρ, c and d can be chosen.

The proposed scheme enables users to manage the future distribution of their shared contents, so that they gradually get corrupted. An obvious question for the scheme is how to handle previous copies of the content. One might have obtained a previous copy of the content, for example one has stored it on the first access, when it was possible right. Yet, the same problem exists in the case of the expiration day

erasure policy as well. The only way to address to this problem is through embedding digital watermarks in the content, so either the applications that process the content apply the proper policy, either the leaked copies can be traced back to their source, through not just sharing a watermarked version of the content but fingerprinting it for each user that access it. Of course in this case, since we may have alternations on the original content due to the added noise, the embedded watermarks should be robust against such added noise.

VI. Applications

One very obvious application of the proposed scheme is in social networks. Currently social networks count millions of subscribers all over the globe, people post vast amount of information about themselves. Quite recently people have started thinking about what this information can cause them in the future. It is known that there are many awkward photos, videos, posts regarding past events or relationships. By using the proposed scheme one could set how important each post is in order to set the appropriate error tolerance. By doing so the server may decide to apply the error distortion with the pass of time or with the usage. Depending on the type of file of the shared content and how tolerant it is to faults, the added noise may result in different outcomes. For example in text messages or raw image and sound files, the added noise might end up to something partly usable for some time, or usable with minor distortions.

If we would like to move to another form of interacting with Social Networks (SNs), where the content is just distributed by them and the user is in total control of his data, then we can have several hosting plans for the shared content, as illustrated in figure 2. Since public key encryption schemes enable us to have two different keys the public and the secret one, one key can be used for encryption and the other for decryption. So in the proposed hosting plans, the keys, whether they are private or secret, cannot be retrieved by the SN. The registered users may exchange their keys through other channels, distinct from the SN they are registered. Therefore, they can implement their desired policy without any undesirable breaches from the SN.

Depending on the privacy policy that Alice wants to implement on the content we may have the following policies. Alice sends to the Social Network m_1, sharing it as a public available information. Message m_2 is intended to be accessed only by Bob so Alice sends to SN, $E_{Pub_B}(m_2)$. If she wants to send message m_3 to group of recipients and this information to be accessed only by members of the group, Alice may use a group-oriented encryption algorithm, like [7], and send $E_{Priv_{A_G}}(m_3)$ to the SN. If she wants the content of message m_4 to deteriorate with the pass of time and to be only available to Bob, Alice sends to $E_{PrivC_A}(m_4)$ SN and a weight w, letting the SN know how important this information is and what the corrosion rate will be. Alice will also have to send Bob $c = E_{Pub_B}(PubC_A)$. The weight w is going to determine the error probability hence the expected weight of the error vector of each update.

Another probable application involves publication of minor law offenses and mugshots. More and more minor law offenders face the problem of being cast away from job interviews due to published track records on the Internet, or being socially ridiculed from their published mugshots. Implementing such schemes will allow a smoother re-socialization of these people, without of course losing this information. Depending on the type of offense and if its recurrence, the amount of error correction and applied noise can be redefined to favor or punish accordingly.

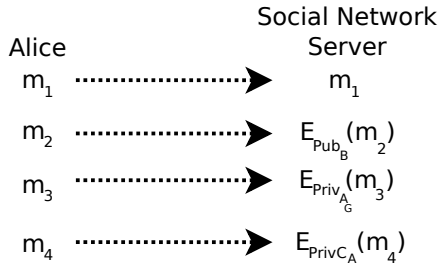

Figure 2. Sharing content on a Social Network.

Apart from privacy centric applications, the proposed model can be applied in Digital Rights Management (DRM). Users can have full access to particular digital content, which deteriorates time after time so that after several uses or after a period of time, it cannot be accessed anymore.

VII. Conclusions

Current mediums and policies have the tendency of storing information forever. This work presents a novel method to provide digital oblivion, which greatly differs from the usual policy, where the data have an expiration day after which they are removed. In our proposal the information corrodes by the pass of time, by gradually adding noise.

In order to create a more trustworthy environment for the users, our plans for future work involve moving to a distributed and server-less model, using a P2P architecture, therefore users privacy will not depend on the current prerequisite fair server. Moreover trying to embed a co-privacy model in them [2] will force users to protect the privacy of the others in order to protect their own as well. Therefore instead of depending on one server to act fairly, the users will help others to maintain their privacy, as their own depends on them.

Disclaimer and acknowledgments

The author is with the UNESCO Chair in Data Privacy, but he is solely responsible for the views expressed in this paper, which do not necessarily reflect the position of UNESCO nor commit that organization. This work was partly supported by the Spanish Government through project CONSOLIDER INGENIO 2010 CSD2007-00004 “ARES”.

References

[1] Commission Proposal for a Regulation of the European Parliament and of the Council, art. 4(2), COM (2012) 11 final (Jan. 25, 2012), available at http://ec.europa.eu/justice/data-protection/document/review2012/com_2012_11_en.pdf.

[2] J. Domingo-Ferrer, “Rational Enforcement of Digital Oblivion”, 4th International Workshop on Privacy and Anonymity in the Information Society (PAIS 2011) collocate with EDBT/ICDT, Uppsala, Sweden, In Proceedings of the 4th International Workshop on Privacy and Anonymity in the Information Society, ISBN: 978-1-4503-0528, Mar 2011

[3] J. R. Anderson, “Learning and memory: An integrated approach”, New York, John Wiley & Sons, 2000.

[4] V. Mayer-Schönberger, “The Virtue of Forgetting in the Digital Age”, Princeton and Oxford: Princeton University Press, 2009.

[5] R. Ostrovsky, O. Pandey, and A. Sahai, “Private locally decodable codes”, ICALP 2007: 387-398.

[6] B. Hemenway and R. Ostrovsky, “Public key encryption which is simultaneously a locally-decodable error correcting code”, Electronic Colloquium on Computational Complexity (ECCC) 14(021), 2007.

[7] B. Hemenway and R. Ostrovsky, “Public-Key Locally-Decodable Codes”, In CRYPTO 08, pages 126-143, 2008.

[8] T. R. N. Rao, “Joint encryption and error correction schemes”, SIGARCH Comput. Archit. News, 12(3), 1984, pp. 240–241. 20.

[9] S. C. Kak, “Joint encryption and error correction coding”, IEEE Conference on Security and Privacy, 1983, pp. 55-60.

[10] R. J. McEliece, “A public-key cryptosystem based on algebraic coding theory”, Deep Space Network Progress Report, Nos. 42-44, Jet Propulsion Labs, Pasadena, CA. 1978, pp. 114-116.

[11] S. Goldwasser, S. Micali, “Probabilistic encryption”, Journal of Computer and System Sciences 28 (2): 270–299, 1984.

[12] O. Goldreich, D. Ron, and M. Sudan, “Chinese remaindering with errors”, In STOC ’99: Proceedings of the thirty-first annual ACM symposium on Theory of computing, pages 225–234, New York, NY, USA, 1999. ACM Press.

[13] T. Hwang, “Cryptosystem for group oriented cryptography”, Proceedings of the workshop on the theory and application of cryptographic techniques on Advances in cryptology, pp 352-360, 1991.

[14] S. Yekhanin, “Locally decodable codes”, Foundations and trends in theoretical computer science, 2010.

[15] C. Cachin, S. Micali, and M. Stadler, “Computationally private information retrieval with polylogarithmic communication”, In Advances in Cryptology: EUROCRYPT ‘99, volume 1592 of Lecture Notes in Computer Science, pages 402–414. Springer Verlag, 1999.

[16] C. Gentry and Z. Ramzan, “Single-database private information retrieval with constant communication rate”, In Automata, Languages and Programming, volume 3580 of Lecture Notes in Computer Science, pages 803–815. Springer Berlin / Heidelberg, 2005.

[17] T.R.N. Rao and K.H. Nam, “Private-Key Algebraic-Code Encryptions”, IEEE Trans. Info. Theory, pp. 829–833, 1989.

[18] T. Hwang and T.R.N. Rao, “Secret Error-Correcting Codes (SECC)”, Proc. Crypto’88, pp. 540–563, 1988.

[19] Kencheng Zeng, Chung-Huang Yang and T.R.N. Rao, “Cryptanalysis of the Hwang-Rao Secret Error-Correcting Code Schemes”, In Informations and communications security, Volume 2229/2001, 419-428, 2001.

[20] Qi Chai, Guang Gong, “Differential Cryptanalysis of Two Joint Encryption and Error Correction Schemes”, Global Telecommunications Conference (GLOBECOM 2011), 2011 IEEE , pp.1-6, 5-9 Dec. 2011.

[21] H. M. Sun, “Private-key cryptosystem based on burst-error-correcting codes”, Electronics Letters, vol. 33, November 1997, pp. 2035–1036.

[22] S. Goldwasser, S. Micali, “Probabilistic encryption and how to play mental poker keeping secret all partial information”, Proc. 14th Symposium on Theory of Computing: 365–377, 1982.