How to Store some Secrets

Reto Koenig[1,2], Rolf Haenni[1]
[1]*Bern University of Applied Sciences, CH-2501 Biel, Switzerland*
[2]*University of Fribourg, CH-1700 Fribourg, Switzerland*
{*reto.koenig,rolf.haenni*}*@bfh.ch*

Keywords: Cryptography, Symmetric Encryption, Polynomials over Finite Fields, Password Manager

Abstract: This paper introduces a special type of symmetric cryptosystem called multi-encryption scheme. It allows users to encrypt multiple plaintexts into a single ciphertext. Each plaintext is protected with its own secret key, meaning that they can be decrypted individually by applying the decryption function with the corresponding key to the ciphertext. Compared to encrypting the ciphertexts one-by-one using a standard symmetric cryptosystem, the main advantage of using a multi-encryption scheme is the no-search property, which guarantees that knowing the key is sufficient for decrypting a single plaintext. We show how to construct a multi-encryption scheme based on polynomials over finite fields. A possible application area is coercion-resistant electronic voting. To ensure a strong form of privacy, voters are equipped with multiple fake credentials, which are indistinguishable from the proper one. While theoretically sound, this requires a voter to perfectly recall multiple lengthy random numbers, and to know which of them is the proper one. To ensure 100% recall, users need to manage these numbers and keep them secret. A multi-encryption scheme is an elegant solution for this problem.

1 INTRODUCTION

In strong cryptographic systems, keys are long random numbers, very long and very random, so that it is hard for a human user to reproduce all these keys from memory (Brown et al., 2004; Florêncio and Herley, 2007). Various approaches addressing this problem have been introduced. One example is keeping a record of the keys, either in plaintext or as an encrypted *password vault* using an easy-to-remember password or passphrase. Clearly, the plaintext variant is not desirable. However, encrypting multiple secret keys in a single password vault (e.g., KEEPASS) is also not ideal: either the password vault informs the user whether or not the entered password is correct (enabling an offline attack), or the password vault produces values for incorrect passwords (meaning the user cannot be sure if the correct values came out).

1.1 Contribution

In this paper, we present a simple multi-encryption scheme satisfying the needs of both sides, persons and cryptographic systems providing the property of perfect secrecy. In particular, it is able to encrypt any amount of pre-chosen high-entropy plaintexts of any means (e.g., private keys, e-voting credentials, different but similar looking PUK numbers, etc.) in a single ciphertext, using an individual and user-chosen low-entropy key for every plaintext. The decryption of such a plaintext is achieved very efficiently by applying only the corresponding key.

Below, we give an informal list of minimal basic requirements a multi-encryption ciphertext must satisfy to be useful in practical applications.

- **No search.** Applying a genuine key to a multi-encryption ciphertext directly reveals the corresponding plaintext, without the need of further discrimination amongst the plaintexts. This very special feature states a key property of multi-encryption schemes and is in accordance to the works of Stajano (Stajano, 2011) allowing humans to deal with high-entropy data serving as secret-keying material for some cryptographic system.
- **Indistinguishability of keys.** False keys must return plaintexts indistinguishable from actual plaintexts. This ensures that an adversary cannot distinguish false keys from genuine keys. This property must hold true even if the plaintext alphabet is limited.
- **Independence of keys.** Knowledge of one or sev-

eral keys of a given ciphertext must not reveal any information about any remaining keys or plaintexts.

- **Reusability of keys.** It must be possible to reuse the keys used within one ciphertext in other ciphertexts without any loss of security. Hence it must be secure against differential cryptographic analyzes.

1.2 Example Applications

The aim of the following examples is to demonstrate the potential of such multi-encryption schemes and their usage allowing humans to manage the rising demands of modern e-society, requesting more and more high-entropy keying material to guarantee security and privacy.

Coercion-Resistant E-Voting. Coercion-resistant e-voting schemes, such as (Koenig et al., 2011), require the voter to remember several high entropy credentials. These credentials are used in order to allow a voter to vote. However, there are two flavors of credentials. The 'good one' is silently accepted while the 'dummy one' is silently rejected by the voting scheme. The good credential is required to allow an eligible voter to cast the vote. The dummy credentials are required to fake the cast of a vote. But all credentials look the same. In order to stay coercion resistant, the voter is not allowed to write the credentials down, or to mark them accordingly. Moreover, no one else than the voter is allowed to know the amount of personal credentials. All these extremely user-unfriendly requirements have mainly motivated the multi-encryption scheme. The scheme now allows the voter to create a ciphertext, where all credentials can be placed securely. The multi-encryption ciphertext will not unveil the exact amount of credentials that were placed within. Furthermore, the voter can use any input as key, serving as a hint for the voter only, in order to recover the desired credential (e.g. think of the name of a fruit for the good credential and the name of vegetables for the dummy credentials). However, any input will always result in a plausible credential. The ciphertext does not leak the true credentials it carries. This renders the ciphertext unchallengeable neither for any credential, nor for the amount of credentials. After entering the correct key, the according credential will be presented without having to search for it or having to mark it. Only the use of a multi-encryption scheme renders such e-voting schemes usable by human end-users.

Remember the PUKs. Todays mobile computing requires the knowledge of PUKs (personal unblocking key) for SIM-cards in order to use them in the GSM-network. These PUKs are composed of an arbitrary immutable eight digit decimal number. Even though PUKs can not be changed by the user, the user needs to remember them precisely. Using a multi-encryption scheme, all needed PUKs can be encrypted in a single ciphertext, each PUK recoverable by applying its appropriate freely selectable decryption key. So the PUK of mom's SIM-card could be released by the key "Mama's smarty", whereas the PUK of father's mobile SIM-card would be accessible by "Daddies 2nd dearest". However, entering any other key would always result in a plausible but wrong PUK code. This way no adversary is able to challenge a given ciphertext, as every outcome is a plausible outcome. Without the SIM-card, the multi-encryption ciphertext does not leak the true PUK codes at all. In possession of the SIM-card, the adversary will be given ten challenges to unlock the system. If the adversary does not succeed, the SIM-card is rendered unusable. So even if keys with somewhat low entropy are used as encryption keys, the adversary would only get a very low chance to guess the correct PUK using the ciphertext. This implies that the ciphertext can be made public without unveiling any information to a potential adversary.

Context Aware Decryption During their bachelor thesis, two of our students implemented a multi-encryption scheme as an application on a smartphone. By enriching the key space with sensor data provided by the smartphone, they were able to demonstrate a location-aware access control system, where the correct credential was provided only if the correct geographical location and the BSSIDs of some WLAN access points were entered. Any other geographical place would have returned a plausible but wrong credential.

1.3 Paper Overview

In Section 2 we give a formal description of what we consider to be a multi-encryption scheme. We also describe possible modes of application of a general multi-encryption scheme. Then, in Section 3, we present a concrete instance of a symmetric multi-encryption scheme based on polynomial interpolation. Finally, Section 4 summarizes our findings and gives an outlook to future work.

2 MULTI-ENCRYPTION SCHEME

We consider a *plaintext space* $\mathcal{M}$, the set of all possible plaintexts, and a *key space* $\mathcal{K}$, the set of all possible keys. Let $M = (m_1, \ldots, m_n)$, $m_i \in \mathcal{M}$, be a list of $n \geq 1$ (not necessarily distinct) plaintexts and $K = (k_1, \ldots, k_n)$, $k_i \in \mathcal{K}$, an equally long list of distinct keys. Furthermore, let $\mathcal{C}$ be the *ciphertext space*, the set of all possible ciphertexts.

2.1 Deterministic Multi-Encryption Scheme

The two main components of a multi-encryption scheme are functions for encrypting and decrypting plaintexts. Formally, let

$$\mathsf{encrypt} : \mathcal{M}^n \times \mathcal{K}^{(n)} \longrightarrow \mathcal{C}$$

denote the *multi-encryption function*, where $\mathcal{K}^{(n)} \subseteq \mathcal{K}^n$ denotes the set of all proper key lists (those with distinct keys), and

$$\mathsf{decrypt} : \mathcal{C} \times \mathcal{K} \longrightarrow \mathcal{M}$$

the *decryption function*. As a notational convention, we will write the functions with the keys as index, i.e., $c = \mathsf{encrypt}_K(M) \in \mathcal{C}$ for encrypting a list of plaintexts $M \in \mathcal{M}^n$ with a list of distinct keys $K \in \mathcal{K}^{(n)}$ and $m_i = \mathsf{decrypt}_{k_i}(c) \in \mathcal{M}$ for the corresponding decryption with key $k_i \in K$. Clearly, we require

$$\mathsf{decrypt}_{k_i}(\mathsf{encrypt}_K(M)) = m_i$$

to hold for all $1 \leq i \leq n$. We call n the *size* of c.

To render a multi-encryption scheme usable for its intended purposes, we require it to possess the cryptographic properties of a traditional symmetric cryptosystem (Menezes et al., 1996). As a consequence, we require that knowing c does not unveil any information about M (except possibly its length), or in technical terms, that $H(M|c)$, the conditional Shannon entropy of M given c, is equal to $H(M)$. More generally, consider a non-empty subset of indices $I \subseteq \{1, \ldots, n\}$ and corresponding sublists $M_I \subseteq S$ and $K_I \subseteq K$ of length $s = |I|$. Furthermore, let $I\text{-}\mathsf{encrypt} : \mathcal{M}^s \times \mathcal{K}^{(s)} \longrightarrow \mathcal{C}$ be the *partial multi-encryption function* derived from encrypt by choosing m_i and k_i arbitrarily for all $i \notin I$, and let $I\text{-}\mathsf{decrypt} : \mathcal{C} \times \mathcal{K}^{(s)} \longrightarrow \mathcal{M}^m$ be the *extended decryption function* obtained from applying decrypt for each $k_i \in K_I$ in the given order. This clearly implies $I\text{-}\mathsf{decrypt}_{K_I}(I\text{-}\mathsf{encrypt}_{K_I}(M_I)) = M_I$. Therefore, we require that I-encrypt together with I-decrypt satisfies the cryptographic properties of a symmetric cryptosystem for all non-empty subsets $I \subseteq \{1, \ldots, n\}$. In particular, we require that decrypting some plaintexts from c does not disclose any information about the other plaintexts in c, or in technical terms, that $H(M_I|c, M_J) = H(M_I)$ holds for all complementary subsets $I, J \subseteq \{1, \ldots, n\}$. As a consequence, $H(m_i|c) = H(m_i)$ and $H(m_i|c, m_j) = H(m_i)$ must hold individually for every $m_i \in M_I$ and $m_j \in M_J$.

Definition 1. A *multi-encryption scheme* of order n,

$$\Sigma[n] = (\mathcal{M}, \mathcal{K}, \mathcal{C}, \mathsf{encrypt}, \mathsf{decrypt}),$$

consists of a plaintext space $\mathcal{M}$, a key space $\mathcal{K}$, a ciphertext space $\mathcal{C}$, and two functions encrypt (with two arguments of arity n) and decrypt with properties as introduced above.

A multi-encryption scheme cannot be constructed by simply concatenating n symmetric cryptosystems. The difference is subtle but crucial: k_i alone is not sufficient for decrypting the i-th plaintext from a list of ciphertexts $c = (c_1, \ldots, c_n)$. While m_i could easily be retrieved by decrypting c_i, additional knowledge of the position i would be required for selecting c_i from c. However, since (i, k_i) is not an element of the key space, we get a more restrictive setting for decrypting the plaintexts. Note that a multi-encryption scheme is also different from a single symmetric cryptosystem, which is used to encrypt a list of plaintexts $M \in \mathcal{M}^n$ with a single master key $k \in \mathcal{K}$. This is the functionality generally provided by a password vault system. Clearly, classical symmetric cryptosystems and password vault systems are both special cases of a multi-encryption scheme of order $n = 1$.

2.2 Randomized Multi-Encryption Scheme

A multi-encryption scheme according to the above definition is *deterministic*, i.e., encrypting a given list of plaintexts M with a given list of distinct keys K always results in the same ciphertext c. Some applications, however, may require a randomized multi-encryption function. Formally, let R be a *randomization space*, the set of all possible randomizations, and

$$\mathsf{randEncrypt} : \mathcal{M}^n \times \mathcal{R} \times \mathcal{K}^{(n)} \longrightarrow \mathcal{C}$$

the resulting *randomized multi-encryption function* (decrypt remains unchanged). As in the deterministic case, we expect

$$\mathsf{decrypt}_{k_i}(\mathsf{randEncrypt}_K(M, r)) = m_i$$

to hold for all $1 \leq i \leq n$ and $r \in \mathcal{R}$. We also assume the same cryptographic properties to hold with respect to corresponding functions I-randEncrypt and

I-decrypt. Finally, we require that different randomizations distribute corresponding ciphertexts uniformly over $\mathcal{C}$, i.e.,

$$P(\mathsf{randEncrypt}_K(M,r) = \mathsf{randEncrypt}_K(M,r')) = \frac{1}{|\mathcal{C}|}$$

is the probability of getting the same ciphertext for two different randomizations $r \neq r'$.

Definition 2. A *randomized multi-encryption scheme* of order n,

$$\tilde{\Sigma}[n] = (\mathcal{M}, \mathcal{R}, \mathcal{K}, \mathcal{C}, \mathsf{randEncrypt}, \mathsf{decrypt}),$$

consists of a plaintext space $\mathcal{M}$, a randomization space $\mathcal{R}$, a key space $\mathcal{K}$, a ciphertext space $\mathcal{C}$, and two functions randEncrypt (with the first and the last argument of arity n) and decrypt with properties as introduced above.

2.3 Modes of Application

The standard mode of application of a multi-encryption scheme of order n is to encrypt n different plaintexts with n different keys. In this section, we present further options for using a multi-encryption scheme by making the keys and the plaintexts dependent on each other. We provide a non-exhaustive list of modes of application, which may be useful for various purposes. Note that arbitrary combinations of these modes of application are possible.

Disjunctive Keys. To allow a single plaintext m to be accessed redundantly by multiple keys, we construct a ciphertext $c = \mathsf{encrypt}_K(M)$ of size n for

$$K = (k_1, \ldots, k_n) \text{ and } M = (m, \ldots, m),$$

where M contains n copies of the same plaintext m and K the different keys to decrypt it. This can be used for defining one or several backup keys, which allow access to the plaintext in case the master key is lost or forgotten. Another application is the detection of typos by accepting a plaintext only after decrypting it independently from at least two different keys. Please note, that if low-entropy keys only are in use, the ciphertext can be challenged, as several low-entropy keys result in the same plaintext. So, only one low-entropy key is allowed whereas the other keys must be of high-entropy.

Conjunctive Keys. To allow access to a single plaintext m only if several keys are provided, we construct a ciphertext $c = \mathsf{encrypt}_K(M)$ of size $n+1$ for

$$K = (k_1, \ldots, k_n, k) \text{ and } M = (m_1, \ldots, m_n, m),$$

where $k_1, \ldots, k_n$ are the n keys required to decrypt the plaintext, $m_1, \ldots, m_n$ are chosen at random from $\mathcal{M}$, and $k = \psi(m_1, \ldots, m_n) \in \mathcal{K}$ is an additional key obtained from applying some n-ary function $\psi : \mathcal{M}^n \to \mathcal{K}$ to the randomly chosen plaintexts. Note that if ψ is symmetric, then the order of the keys does not matter to decrypt the plaintext. A direct application of this setting is the combination of multiple low-entropy keys (e.g., passwords) into a high-entropy key.

Threshold Keys. By combining the above cases of disjunctive and conjunctive keys, any monotone Boolean function can be constructed. Of particular interest are (t,n)-threshold functions, where at least t (out of n) keys are needed to decrypt the plaintext m. Generally, such threshold functions can be expressed by a disjunction of $\binom{n}{t}$ conjunctions of length t. For $t=2$ and $n=3$, for example, we construct a ciphertext $c = \mathsf{encrypt}_K(M)$ of size $3 + \binom{3}{2} = 6$ for

$$K = (k_1, k_2, k_3, k_{12}, k_{13}, k_{23}),$$
$$S = (m_1, m_2, m_3, m, m, m),$$

where m_1, m_2, m_3 are chosen at random from $\mathcal{M}$, and $k_{ij} = \psi(m_i, m_j)$ are additional keys obtained from applying some binary function $\psi : \mathcal{M} \times \mathcal{M} \to \mathcal{K}$ to all pairs of random plaintexts.

Chains of Plaintext. To decrypt a sequence of plaintexts from a single key k, we construct a ciphertext $c = \mathsf{encrypt}_K(M)$ of size n for

$$K = (k, k_2, \ldots, k_n) \text{ and } M = (m_1, \ldots, m_n),$$

where M contains the plaintexts, and $k_{i+1} = \psi(m_i)$, $1 \leq i \leq n-1$, are $n-1$ additional keys obtained from applying some unitary function $\psi : \mathcal{M} \to \mathcal{K}$ to the plaintexts (for $\mathcal{K} = \mathcal{M}$, ψ may simply be the identity function). To mark the end of such a chain of plaintexts, a special marker $m_0 \in \mathcal{M}$ must be attached as an additional plaintext to M. Note that by applying this technique multiple times in parallel, we can construct a ciphertext containing several independent chains of plaintexts. More generally, we can construct arbitrary directed graphs of interlinked plaintexts (with multiple sources or sinks, loops, etc.), whose nodes have an outdegree of at most 1.

Non-Confidential Plaintexts. To enhance a ciphertext of size n with r non-confidential metadata, say $m'_1, \ldots, m'_r \in \mathcal{M}$, we define r additional keys $id_i \in \mathcal{K}$, which serve as identifiers for corresponding values m'_i. Therefore,

$$K = (k_1, \ldots, k_n, id_1, \ldots, id_r),$$
$$S = (m_1, \ldots, m_n, m'_1, \ldots, m'_r),$$

are the enhanced lists of keys and plaintexts, from which the ciphertext $c = \mathsf{encrypt}_K(M)$ of size $n+r$ is constructed. This can be used for adding metadata about the ciphertext (e.g., author, date of creation, description, notes, etc.), the keys (e.g., recovery hints), or the plaintexts (e.g., application domain, expiration date, etc.) to the ciphertext.

3 MULTI-ENCRYPTION SCHEME BASED ON POLYNOMIALS OVER FINITE FIELDS

In this section, we introduce a concrete instance of a randomized multi-encryption scheme $\tilde{\Sigma}[n] = (\mathcal{M}, \mathcal{R}, \mathcal{K}, \mathcal{C}, \mathsf{randEncrypt}, \mathsf{decrypt})$. The general idea of our approach corresponds to the Reed-Solomon coding scheme (Reed and Solomon, 1960), which is based polynomial interpolation over a finite field. To create a ciphertext, randEncrypt will therefore construct a polynomial that contains a point for each of the n plaintexts, and to decrypt a single plaintext from the ciphertext, decrypt simply applies the polynomial on the corresponding key.

3.1 Construction

Let p be a large prime number and $\mathbb{Z}_p$ the corresponding finite field of integers modulo p. The set of all possible polynomials over $\mathbb{Z}_p$ is denoted by $\mathbb{F}_p[x]$ and serves as our ciphertext space $\mathcal{C}$. To apply the Lagrange interpolation formula (or any other interpolation method) directly on points (k_i, m_i), we start with the simplifying assumption of $\mathcal{K} = \mathcal{M} = \mathcal{R} = \mathbb{Z}_p$. The resulting polynomial $c(x) = \mathsf{Lagrange}((k_1,m_1),\ldots,(k_n,m_n)) \in \mathbb{F}_p[x]$ is then the ciphertext, from which every $m_i = c(k_i)$ is easily retrieved. Unfortunately, there are at least two major problems with this basic solution:

- For $n = 1$ or, more generally, if $m_i = m_1$ holds for all plaintexts m_i, then the polynomial degenerates into a straight horizontal line $c(x) = m_1$. To avoid such trivial cases, we suggest that $(0,0)$ is added to the list of given interpolation points (k_i, m_i) and that the value 0 is removed from both sets $\mathcal{K}$ and $\mathcal{M}$.
- The resulting polynomial $c(x)$ is of very low order ($\leq n-1$), which makes it easy to factorize (Berlekamp, 1970), (Kaltofen and Shoup, 1998). As a countermeasure, we suggest combining the interpolation polynomial with a monomial x^r of very high random order $r \geq 2^u$, where u is a security parameter. For this to work, we must adjust the interpolation points from (k_i, m_i) to (k_i, m'_i) by $m'_i = m_i - k_i^r$. Then the interpolation yields $c'(x) = \mathsf{Lagrange}((k_1,m'_1),\ldots,(k_n,m'_n))$, from which we obtain the ciphertext $c(x) = x^r + c'(x)$. This way we get a very sparse polynomial of very high order, a combination allowing an easy evaluation but a hard inversion.

By combining the two suggested enhancements of the basic solution, we obtain the following randomized multi-encryption and decryption functions:

```
Function randEncrypt_K(M, r)
Input: M = (m_1, ..., m_n), K = (k_1, ..., k_n), r
begin
    k_0 ← 0, m'_0 ← 0
    for i ← 1 to n do
        m'_i ← m_i − k_i^r
    c(x) ← x^r + Lagrange((k_0, m'_0), ..., (k_n, m'_n))
    return c
```

Note that this solution bears a small risk of obtaining

```
Function decrypt_{k_i}(c)
Input: c, k_i
begin
    return c(k_i)
```

a degenerated polynomial, since it may happen that $m_i = k_i^r$ for all $1 \leq i \leq n$. As a countermeasure, we further restrict $\mathcal{K}$ to a large sub-group $G_q \subset \mathbb{Z}_p^*$ of order q (e.g., by selecting a safe prime $p = 2q+1$) and $\mathcal{M}$ to the complement $\overline{G}_q = \mathbb{Z}_p^* \setminus G_q$. This implies $m_i \neq k_i^r$ and therefore $m'_i \neq 0$. Note that this solution also imposes a upper limit $r < q$ for the randomization, i.e., the remaining randomization space for a given security parameter u is $\mathcal{R} = [2^u, q-1]$.

In the most general case, where $\mathcal{K}$, $\mathcal{M}$, and $\mathcal{R}$ are general sets, we can apply the above randomized multi-encryption function only after mapping the keys into G_q, the plaintexts into $\overline{G}_q$, and the randomization into $[2^u, q-1]$. Let $\kappa : \mathcal{K} \to G_q$, $\sigma : \mathcal{M} \to \overline{G}_q$, and $\rho : \mathcal{R} \to [2^u, q-1]$ be such mappings. Note that σ must be invertible, since decrypting a plaintext requires applying σ^{-1} to $\sigma(m_i)$, and this imposes a general restriction $|\mathcal{M}| \leq |\overline{G}_q|$ on the size of the plaintext space ($|\mathcal{M}| \leq q$ for $p = 2q+1$). To preserve the distinctness of the keys in $K = (k_1, \ldots, k_n)$, similar considerations apply to κ. However, instead of requiring κ being injective and thus imposing the restriction $|\mathcal{K}| \leq q$ on the size of the key space, we only assume that κ is collision-resistant. With this, no more restrictions apply to the key space, i.e., $\mathcal{K}$ may even be an infinite set. To avoid collisions entirely, consider a parameterized collision-resistant function $\kappa : \mathcal{K} \times \mathbb{N} \to G_q$, for which we choose each time a parameter $z \in \mathbb{N}$ such that $\kappa(k_i, z) \neq \kappa(k_j, z)$ holds for all $i \neq j$. In that case, z becomes part of the ciphertext.

3.2 Discussion

Security. We will first discuss the security of a multi-encryption scheme of order $n = 1$ and then augment our arguments inductively to bigger size. For $n = 1$, the resulting interpolation polynomial can be written as:

$$c(x) = x^r + c'(x) = x^r + \frac{m_1 - k_1^r}{k_1} x.$$

Knowing m_1 allows the adversary to learn k_1 by solving the equation $c(x) = m_1$, or in other words, by finding the roots of the polynomial $c(x) - m_1 = x^r + \frac{m_1 - k_1^r}{k_1} x - m_1$. The adversary may either try an exhaustive search or factorizing the polynomial, but both options are considered to be hard for a sufficiently large key space and a very high order $r \geq 2^u$. Exactly the same argument holds for each of the plaintexts in the more general case $n > 1$.

Now suppose the adversary learns one key k_i from a ciphertext of size $n > 1$. Would this reveal any information about the other keys or plaintexts? In that case, the adversary is able to reduce the order of the polynomial $c(x)$ by one (from r to $r - 1$, by dividing $c(x) - m_i$ by $x - k_i$), but this means that all other keys and plaintexts remain completely hidden under the above assumptions of a sufficiently large key space and a very high order.

Efficiency. If we consider the field operations of $\mathbb{Z}_p$, i.e., addition, subtraction, multiplication, and division modulo p, as primitive operations, we need $O(n \log r + n^2)$ many operations to compute $\mathsf{randEncrypt}_K(M, r)$, i.e., $O(\log r)$ operations for each value k_i^r and $O(n^2)$ for polynomial interpolation. To compute $\mathsf{decrypt}_{k_i}(c)$, we require $O(\log r + n \log n)$ operations, i.e., $O(\log r)$ operations for the monomial k_i^r and $O(\log n)$ operations for every low-order term in $c'(x)$. Therefore, if we treat $\log r$ as a constant (its lower bound is u and it upper bound is $\log q$), we can summarize the running times by saying that constructing a ciphertext is quadratic and decrypting a plaintext is quasilinear in n. Note that n will be rather small in most applications.

4 CONCLUSION

A multi-encryption scheme is a new cryptographic concept for protecting the confidentiality of multiple plaintexts in a single ciphertext. It is similar to a symmetric cryptosystem, but the ability to decrypt the plaintexts individually using respective keys adds a great amount of flexibility. We have shown in this paper some of these possibilities. One of the main applications of a ciphertext is to facilitate human interaction with complex cryptographic protocols, which require users to remember multiple lengthy random numbers. The proposed realization of a multi-encryption scheme based on polynomial interpolation offers the necessary security and efficiency. A functional prototype written for a smartphone has already proven its usability. The inclusion of sensor data in the key space even allows automatic decryption based on a given context.

Future work will focus on a generic construction method based on existing cryptosystems, deterministic or asymmetric multi-encryption schemes, a more profound security analysis.

REFERENCES

Berlekamp, E. R. (1970). Factoring polynomials over large finite fields. *Mathematics of Computation*, 24:713–735.

Brown, A. S., Bracken, E., Zoccoli, S., and Douglas, K. (2004). Generating and remembering passwords. *Applied Cognitive Psychology*, 18(6):641–651.

Florêncio, D. and Herley, C. (2007). A large-scale study of web password habits. In Patel-Schneider, P. F. and Shenoy, P., editors, *WWW'07, 16th International Conference on World Wide Web*, pages 657–666, Banff, Canada.

Kaltofen, E. and Shoup, V. (1998). Subquadratic-time factoring of polynomials over finite fields. In *Math. Comp*, pages 398–406.

Koenig, R., Haenni, R., and Fischli, S. (2011). Preventing board flooding attacks in coercion-resistant electronic voting schemes. In Camenisch, J., Fischer-Hübner, S., Murayama, Y., Portmann, A., and Rieder, C., editors, *SEC'11, 26th IFIP International Information Security Conference*, volume 354, pages 116–127, Lucerne, Switzerland.

Menezes, A. J., van Oorschot, P. C., and Vanstone, S. A. (1996). *Handbook of Applied Cryptography*. CRC Press, Boca Raton, USA.

Reed, I. S. and Solomon, G. (1960). Polynomial codes over certain finite fields. *Journal of the Society for Industrial and Applied Mathematics*, 8(2):300–304.

Stajano, F. (2011). Pico: No more passwords! In *IWSP'11, 19th Security Protocols Workshop*, Cambridge, U.K.