PROCEEDINGS OF THE ROMANIAN ACADEMY

Series A:

Mathematics, Physics, Technical Sciences, Information Science

Volume 14 *Special issue 2013*

CONTENTS

INFORMATION SCIENCE

THE PUBLISHING HOUSE
OF THE ROMANIAN ACADEMY

PROCEEDINGS OF THE ROMANIAN ACADEMY, Series A,
Volume 14, Special Issue 2013, pp. 269–285

A FULLY DYNAMIC UNIVERSAL ACCUMULATOR

Atefeh MASHATAN and Serge VAUDENAY

EPFL, Lausanne, Switzerland
http://lasec.epfl.ch

A dynamic universal accumulator is an accumulator that allows one to efficiently compute both membership and nonmembership witnesses in a dynamic way. It was first defined and instantiated by Li *et al.*, based on the Strong RSA problem, building on the dynamic accumulator of Camenisch and Lysyanskaya. We revisit their construction and show that it does not provide efficient witness computation in certain cases and, thus, is only achieving the status of a *partially dynamic* universal accumulator. In particular, their scheme is not equipped with an efficient mechanism to produce non-membership witnesses for a new element, whether a newly deleted element or an element which occurs for the first time.
We construct the first *fully dynamic* universal accumulator based on the Strong RSA assumption, building upon the construction of Li *et al.*, by providing a new proof structure for the non-membership witnesses. In a fully dynamic universal accumulator, we require that not only one can always create a membership witness without having to use the accumulated set for a newly added element, but also one can always create non-membership witnesses for a new element, whether a newly deleted element or an element which occurs for the first time, *i.e.*, a newcomer who is not a member, without using the accumulated set.

Key words: Dynamic Accumulators, Universal Accumulators.

1. INTRODUCTION

Cryptographic accumulators allow us to encapsulate a large number of elements in a single short *accumulator* along with short *witnesses* that can be used for proving whether or not an element has been accumulated. The notion of cryptographic accumulators was first introduced by Benaloh and de Mare [1] and further pursued by many researchers as they come very practical in many scenarios such as anonymous credential systems and group signatures, see for example [9, 8, 4], and that they can be instantiated based on a variety of techniques and hardness assumptions, for instance, the strong RSA assumption, bilinear maps, the Decisional Diffie-Hellman assumption, and one-way hash functions.

We are now going to focus on a number of schemes which are based on the Strong RSA assumption and they were built one after the other in an evolutionary process. Barić and Pfitzmann [2] followed the work of Benaloh and de Mare [1] and introduced collision-free accumulators. This scheme provided membership proofs. Later, Camenisch and Lysyanskaya [5] augmented the latter work and proposed a dynamic accumulator, in which elements can be efficiently added to and removed from the set of accumulated values. Finally, Li *et al.* [7] built their scheme based on the proposal of Camenisch and Lysyanskaya [5] and introduced universal accumulators in which there is a witness, whether a member or not, for every elements in the input domain. (See [6] for a survey.) Although Li *et al.* promise to provide efficient non-membership proofs, we will see that the structure of the witness fails to offer efficient dynamic proof computation for certain elements and, hence, achieves the desired dynamism only *partially*. In a *fully dynamic* universal accumulator, we require that not only one can always create a membership witness without having to use the accumulated set for a newly added element, but also one can always create non-membership witnesses for a new element, whether a newly deleted element or an element which occurs for the first time, *i.e.*, a newcomer who is not a member, without using the accumulated set.

Although accumulators are not so new elements in cryptographic schemes, formal security definitions and classifications on different requirements have not been adequately dealt with. The literature often fails to

provide exact correctness or security definitions for different classes of accumulators and there have been several security notion proposed. We focus on the strongest security notion, considering a powerful adversary who can invoke the authority with polynomially many accumulator initiations. This notion is referred to as the *Chosen Element Attack* model, in the literature [11]. Informally, a polynomially bounded adversary interacts with the authority who maintains the accumulators. The adversary invokes the authority to initiate a polynomial number of accumulators and make changes to them according to the adversary's instructions on what element to add or delete. Finally, the adversary chooses an element and an accumulator and produces a witness. The adversary wins if the witness proves that the chosen element is not a member when in fact it is, or it proves that the chosen element is a member when in fact it is not.

Accumulators have proven to be a very strong mathematical tool with applications in a variety of privacy preserving technologies where there is a desire to represent a set of elements in a compact form, for example, certificate revocation schemes, anonymous credential systems, and group signatures. In particular, fully dynamic universal accumulators can come handy in a variety of real-life scenarios. For example, consider the set of people who have a medical condition that allows them to benefit from some discount medication, but denies them the access to certain areas, such as swimming pools. These people should be able to efficiently prove their membership at a pharmacy and everyone else should be able to show their nonmembership when entering a swimming pool, for example.

It is known that *batch updates* cannot be done [3]. This means that updating a (non)membership proof without the secret key requires to go through all the accumulator updates.

Our contributions.

In this paper, we first point out the lack of efficiency in the dynamic updating process of the dynamic universal accumulator of Li *et al.* [7], where the authority has to go through the already accumulated set to create non-membership witnesses for certain members, namely newly considered values which are not members and newly deleted members, defying the claim that the scheme provides efficient non-membership proofs in all cases.

Moreover, we introduce the notion of *weak* dynamic accumulators, a special case of dynamic accumulators where the only operation is addition and the elements can dynamically be added to the accumulator. Hence, a dynamic accumulator is trivially a weak dynamic accumulator. Further, we present a generic transformation from a weak dynamic accumulator with a domain having a certain structure, *e.g.*, the domain being a set of odd primes, to a weak dynamic accumulator with a domain of arbitrary form, *e.g.*, a subset of $\{0,1\}^*$.

Furthermore, we formally define what we require from a *fully* dynamic universal accumulator and instantiate the first such scheme based on the Strong RSA assumption and a weak dynamic accumulator with an arbitrary domain. This instantiation builds on the previous schemes based on the same hardness assumption by keeping the structure of the membership proofs, due to Camenisch and Lysyanskaya [5], but providing a new structure for the non-membership proofs. This property, *i.e.*, being *fully* dynamic, comes at a price. Our accumulators are a bit larger. However, as it is more efficient when introducing new elements compared to previously introduced partially dynamic accumulators, it achieves the same level of efficiency as the set of accumulated values is growing. Moreover, the efficacy of the new structure of non-membership proofs allows the authority to perform batch updates, a desired property that had not been achieved successfully so far.

Structure of the paper.

The rest of the paper is organized as follows. Section 2 is dedicated to briefly describing, notations, definitions, different classes of already existing accumulators, and the particular variant of the dynamic accumulator of Camenisch and Lysyanskaya [5] due to Li *et al.* [7]. In Section 3, we define the notion of Weak Dynamic Accumulators and present a generic transformation to obtain a Weak Dynamic Accumulators with arbitrary domain. Finally, Section 4 is devoted to defining Fully Dynamic Universal Accumulators followed by an instantiation whose security is based on the strong RSA assumption. Last but not least, we wrap up with some concluding remarks in Section 5.

2. PRELIMINARIES

In this section, we list definitions, notations, and the building blocks which will be used to construct and analyze our scheme in the following section.

Throughout this paper, we use the expression $y \leftarrow A(x)$ to mean that y is the output of algorithm A running on input x. An algorithm is said to have polynomial running time, if its running time can be expressed as a polynomial in the size of its inputs. If X denotes a set, $|X|$ denotes its cardinality and $x \in_R X$ expresses that x is chosen from X according to the uniform distribution. If X and Y are sets, then $X \setminus Y$ denotes the set of elements in X, but not in Y. For convenience, we also use $X + \{x\}$ and $X - \{x\}$ when an element x is being added in or deleted from a set X. We also use the classical notion of a negligible function: $f : N \to R$ is said to be negligible in k if for any positive polynomial function $p(.)$ there exists a k_0 such that if $k \geq k_0$, then $f(k) < 1/p(k)$.

The strong RSA assumption states that given an RSA modulus n and a random x drawn from $\mathbb{Z}_n^*$, it is infeasible to find $e > 1$ and $y \in \mathbb{Z}_n^*$ such that $y^e = x \bmod n$.

2.1. Evolution of Cryptographic Accumulators

In this section, we illustrate the evolution of the cryptographic accumulators in the literature. There are different notions of security used in the literature and, due to lack of space, we only focus on the strongest notion, sometimes referred to as the *Chosen Element Attack* model. As for the notion of correctness of an accumulator, one requires that correctly accumulated values have verifying witnesses, regardless of the type of accumulator. The literature has often stopped here and has failed to provide a more precise definition of correctness. We will provide the first formal definition of correctness that can be applied to several categories of accumulators with different functionalities.

There is an authority who initiates and maintains the accumulator and interacts with other participants. The authority generates the secret and public keys and keeps a state including the keys, the accumulated value, the set of elements which are accumulated. The authority delivers the proofs to the participants. In the security definition, the adversary asks the authority to provide certain proofs in the form of an oracle.

The definitions below are mostly gathered by Wang et al. [12], but contain some twists to make them consistent with the following sections.

Definition 1 (Accumulators). *An* accumulator *scheme* Acc*, with a domain P, a set* $X \subseteq P$*, and values* $x \in X$ *to be accumulated, consists of the following algorithms.*

– *A setup probabilistic algorithm* $\mathrm{KeyGen}(1^k) \to (K_s, K_p)$*, where* K_s *is only used by the authority and* K_p *is public.*

– *An algorithm* $\mathrm{AccVal}(K_s, K_p, X) \to c$*, which computes an accumulator value* $\mathbf{c}$*, for the set X, from the keys.*[1]

– *An algorithm* $\mathrm{WitGen}(K_s, c, X, x) \to W$ *to generate a proof of membership for* $x \in X$ *in accumulator* c.

– *A predicate* $\mathrm{Verify}(K_p, c, x, W)$ *to check a proof.*

One problem with this primitive is the lack of any possibility to dynamically update a set X. Ideally, we would like to insert or delete members of X and update the accumulator c accordingly. To make it efficient we want all values to have a length which only depends on k and not on the cardinality of X. So, we amend the definition by introducing the UpdEle and UpdWit algorithms. This defines the notion of *Dynamic Accumulator* (see Def. 11 in Appendix).

[1] For some constructions, K_s is not used by AccVal.

The accumulator does not consider proofs of non-membership. This is the next desired functionality to have. This is done with the notion of *Universal Accumulator* (see Def. 12 in Appendix).

A proof W for x is said to be *valid* with respect to (X,c) if and only if the predicate $\mathrm{Verify}(K_p,c,x,W)$ holds. Moreover, a proof W for x is said to be *coherent* with respect to (X,c) if and only if it is valid and $\mathrm{IsMem}(W)$ is equivalent to $x \in X$. One problem with this construction is that the authority must figure out whether or not x is a member of X to generate a coherent proof. That is, the information on X cannot be compressed. The lack of dynamism is also a problem.

Finally, we formalize the dynamic universal accumulator notion due to Li *et al.* [7] as follows. The reason why we call it a *partially* dynamic universal accumulator is discussed below. Note that our formalism is a bit more detailed that theirs to be consistent with the rest of our paper.

Definition 2 (Partially Dynamic Universal (PDU) Accumulators). A partially dynamic universal accumulator PDUAcc*, with a domain P, a set* $X \subseteq P$*, and values* $x \in X$ *to be accumulated, consists of the following algorithms.*

– *A setup probabilistic algorithm* $\mathrm{KeyGen}(1^k) \to (K_s,K_p)$*, where* K_s *is only used by the authority and* K_p *is public.*

– *An algorithm* $\mathrm{AccVal}(K_s,K_p,X) \to c$*, which computes an accumulator value* c *of the set X, from the keys.*

– *An algorithm* $\mathrm{UpdEle}(K_s,K_p,c,\mathrm{op},x) \to (c',\mathrm{extra})$*, where* $\mathrm{op}=+$ *or* $\mathrm{op}=-$*, which computes the accumulator* c' *for* $X\mathrm{op}\{x\}$ *from the accumulator* c *for X. When* $\mathrm{op}=+$*, we must have* $x \notin X$ *and we say that* x *is inserted into X. When* $\mathrm{op}=-$*, we must have* $x \in X$ *and we say that* x *is deleted from X. The algorithm also returns some extra information* extra *, which might be needed for dynamic witness update.*

– *An algorithm* $\mathrm{WitGen}(K_s,c,X,x) \to W$ *to generate a proof of membership or non-membership for the value* x *with respect to accumulator* c *of X.*

– *An algorithm* $\mathrm{UpdWit}(K_p,c,c',\mathrm{extra},\mathrm{op},x,W,y) \to W'$ *to generate a proof* W' *for* y *in accumulator* c' *from a proof* W *for* y *in accumulator* c*, where* $\mathrm{UpdEle}(K_s,K_p,c,\mathrm{op},x) \to (c',\mathrm{extra})$*. It must be the case that* $x \neq y$*.*

– *A predicate* $\mathrm{IsMem}(W)$ *telling whether* W *is a proof of membership (true case) or a proof of non-membership (false case).*

– *A predicate* $\mathrm{Verify}(K_p,c,x,W)$ *to check a proof.*

We let X be a subset of P which is initially empty. Every time we run $\mathrm{UpdEle}(.,.,.,\mathrm{op},x)$, we replace X by $X\mathrm{op}\{x\}$. Clearly, for WitGen to generate coherent proofs, $\mathrm{IsMem} \circ \mathrm{WitGen}$ must be a predicate to decide whether an arbitrary x belongs to X or not. Since we want c to have a length which only depends on k (and not on the cardinality of X), it is not possible to require WitGen to generate coherent proofs without X as an input while X has been filled. It is still possible to invoke WitGen to create from X a new proof. Then, we count on UpdWit to update all coherent proofs.

With this notion, UpdWit cannot create a new witness for $x = y$, that is for a value y which has just been added or deleted. This is why we call it *Partially* Dynamic Universal.

2.2. Formalizing the Notions of Correctness and Security for Accumulators

We now describe our notion of correctness than can be applied to several types of accumulators. This definition is our attempt to formalize the notion of correctness which was missing in the literature. One difficulty is that the accumulator interface introduces many options, and that we want to formalize that

whatever sequence of option is selected, the accumulator always keep consistent properties. Here, the sequence of options is arbitrary. We formalize this by introducing an adversary who can select it maliciously.

Intuitively, the notation $c \prec X$ means that c is a correct accumulator value computed for the set X, while $W \vdash (x, X, c, bool)$ means that W is a valid computed proof for x being/not being (depending on the Boolean $bool$) in $c \prec X$.

Definition 3 (Correctness). *Consider a game in which we first run* $\text{KeyGen}(1^k) \to (K_s, K_p)$ *and allow the adversary to take* K_p *and* K_s *and play with the rest of the algorithms available for the respective accumulator, e.g.,* $\text{AccVal}(K_s, K_p, .)$, $\text{MemWitGen}(K_s, ., ., .)$, $\text{NonMemWitGen}(K_s, ., ., .)$, $\text{UpdEle}(K_s, K_p, ., ., \text{op}, .)$, *or* $\text{UpdWit}(K_p, ., ., ., ., \text{op}, ., ., .)$. *We recursively define the relations* $c \prec X$ *and* $W \vdash (x, X, c, bool)$ *by the following conditions:*

– *If the adversary calls* $\text{AccVal}(K_s, K_p, X) \to c$, *then* $c \prec X$.

– *If* $c \prec X$ *and the adversary queries* $\text{UpdEle}(K_s, K_p, c, \text{op}, x) \to (c', \text{extra})$, *then* $c' \prec X \text{op}\, x$.

– *If* $c \prec X$ *and the adversary queries* $\text{WitGen}(K_s, c, X, x) \to W$ *or* $\text{MemWitGen}(K_s, c, X, x) \to W$, *then* $W \vdash (x, X, c, true)$. *If* $c \prec X$ *and the adversary calls* $\text{NonMemWitGen}(K_s, c, X, x) \to W$, *then* $W \vdash (x, X, c, false)$.

– *If* $c \prec X$ *and the adversary called* $\text{UpdEle}(K_s, K_p, c, \text{op}, x) \to (c', \text{extra})$ then $\text{UpdWit}(K_p, c, c', \text{extra}, \text{op}, x, W, y) \to W'$ with $x \neq y$ and $W \vdash (y, X, c, b)$, then $W' \vdash (y, X\text{op}, c', b)$.

The accumulator scheme is said to be correct if for all probabilistic polynomial time adversaries, and for all possible choices of W, x, X and c, we have that $W \vdash (x, X, c, b)$ *implies* $\text{Verify}(K_p, c, x, W)$ *and, in the case of universal accumulators,* $\text{IsMem}(W) \Leftrightarrow x \in X$.

Note that it is unusual to have an adversary in a definition of correctness. This is necessary here as we want to have correctness whatever the history of interactions with the interface.

Next, we describe the Chosen Element Attack scenario [12] when defining security for a PDU accumulator.

Definition 4 (Chosen Element Attack (CEA) Model). *The security of a PDU accumulator is defined in terms of a game, based on a security parameter* k, *played by a polynomially bounded adversary. Firstly,* KeyGen *is run and* K_p *is given to the adversary. Secondly, the adversary selects a polynomially bounded number* ℓ. *There are registers* K_s, X_i, *and* c_i, $i = 1, \ldots, \ell$, *for a secret key and to keep track of* ℓ *sets* X_i *and their accumulator values* c_i. *Initially, all* X_i*'s are empty and* c_i *is set to* $\text{AccVal}(K_s, K_p, X_i)$. *The adversary can then call an* $\text{UpdEle}(K_s, K_p, c_i, ., .)$ *oracle for a selected* i *which updates* X_i *and* c_i *accordingly. It is not allowed to add an* x *to* X_i *when* x *is already in* X_i, *nor is it allowed to delete* x *from* X_i *when* x *is not in* X_i. *The adversary can also call a* $\text{WitGen}(K_s, c_i, X_i, .)$ *oracle for a selected* i *and an* $\text{AccVal}(K_s, K_p, .)$ *oracle which do not update an* X_i *of* c_i. *After making many oracle queries, the adversary ends by producing some* (i, x, W). *The adversary wins if* W *is an incoherent proof for* x *with respect to* X_i *and accumulator* c_i.

Note that when the algorithms are all deterministic, we can always reduce to $\ell = 1$ and remove access to the $\text{AccVal}(K_s, K_p, .)$ oracle. This follows because calling AccVal on the same inputs is going to produce the same outputs. Moreover, the information that the adversary obtains from calling AccVal for the sets X_i can all be simulated by a single set X.

2.3. An Instantiation of Partially Dynamic Universal Accumulators due to Li *et al.*

We now describe a PDU accumulator based on the scheme presented by Li *et al.* [7].

In what follows, we have a domain P of possible values for x with some specific form. A subset $X \subseteq P$ has an accumulator c and public parameters n and g. A proof of non-membership for $x \in P \setminus X$ for c is a tuple (a,d) such that

$$c^a \equiv d^x g \pmod{n}.$$

By writing a Euclidean division $a = a' + qx$, we can easily transform such a proof (a,d) into a new proof (a',d'), with $d' = dc^{-q} \bmod n$, such that $0 \le a' < x$. We refer to

$$[a,d]_x = (a \bmod x, dc^{-\frac{a-(a \bmod x)}{x}} \bmod n)$$

as the "reduced" proof, where the public parameters are implicit. Note that $[a,d]_x$ can be computed without the secret key for any integer a. When a is a rational number, we need the secret key to compute it.

Domain: P is a set of odd prime numbers, e.g., all odd prime numbers up to a given bound B.

KeyGen(1^k): pick two different prime numbers p and q such that $\frac{p-1}{2}$ and $\frac{q-1}{2}$ are both prime and not in P, take $n = pq$, $r = \frac{1}{2}\lambda(n)$, and g an element of order r in $\mathbb{Z}_n^*$. Then, $K_p = (n,g)$ and $K_s = r$.

AccVal (K_p,X): the value of c is defined by

$$c = g^{\prod_{x \in X} x} \bmod n. \tag{1}$$

Note that K_s is not required to compute c.

UpdEle(K_s,K_p,c,op,x): if $\mathrm{op} = +$, we have $c' = c^x \bmod n$ (K_s is not required). If $\mathrm{op} = -$, we have $c' = c^{\frac{1}{x}} \bmod n$ (K_s is required). It returns $\mathrm{extra} = (c, \mathrm{op}, x, c')$. Since it is not allowed to delete a non-member of X, c' can be computed without K_s but using X by applying (1).

WitGen(Ks,c,X,x): if $x \in X$, then $W = (\mathrm{mem}, w)$ with $w = c^{\frac{1}{x}} \bmod n$. This does not require X. It can also be computed without K_s, but using X by observing that

$$w = c^{\prod_{y \in X - \{x\}} y} \bmod n. \tag{2}$$

If $x \notin X$, then $W = (\mathrm{nonmem}, [a,d]_x)$ with $a = \left(\prod_{y \in X} y\right)^{-1}$ and $d = 1$.

Clearly, the final (a,d) is such that

$$a = \left(\prod_{y \in X} y\right)^{-1} \bmod x \quad \textit{and} \quad d = \left(c^a g^{-1}\right)^{\frac{1}{x}} \bmod n. \tag{3}$$

The above computation requires K_s. Below, we give an algorithm to compute $[a,d]_x$ without K_s, but by going through all X members.

UpdWit(K_p,extra, W,y): with $\mathrm{extra} = (c, \mathrm{op}, x, c')$, there are four cases to update the proof for y in X after adding or deleting x:

- W of the form (mem, w) and x just added ($\text{op} = +$): set $W' = (\text{mem}, w')$ with $w' = w^x \bmod n$.
- W of the form (mem, w) and x just deleted ($\text{op} = -$): set $W' = (\text{mem}, w')$ with $w' = w^z c'^{\frac{1-xz}{y}} \bmod n$ and $z = \frac{1}{x} \bmod y$.
- W of the form (nonmem, a, d) and x just added ($\text{op} = +$): set $W' = (\text{nonmem}, [a', d']_y)$ with $a' = az$, $d' = dc^{-a\frac{1-xz}{y}} \bmod n$ and $z = \frac{1}{x} \bmod y$.
- W of the form (nonmem, a, d) and x just deleted ($\text{op} = -$): set $W' = (\text{nonmem}, [a', d]_y)$ with $a' = ax$.

IsMem(W): is true if and only if W is of the form $(\text{mem},.)$.

Verify(K_p, c,x,W): if W is of the form (mem, w), it is true if and only if $c = w^x \bmod n$. If W is of the form (nonmem, a, d), it is true if and only if $c^a \equiv d^x g (\bmod n)$.

We can compute the non-membership proof $[a, d]_y$ for $y \notin X$ by iteratively going through all $x \in X$. For this, we start with $(a, d) = (1,1)$, which is a proof of non-membership for y in the empty set with accumulator $c = 1$. Then, for each $x \in X$ we update $(a, d) \leftarrow \text{UpdWit}(K_p, (c,+,x,c^x \bmod n), (a, d), y)$ and $c \leftarrow c^x \bmod n$.

Note that all algorithms are deterministic here. It can be easily proven that all oracle calls in the security game preserve the relations (1), (2), and (3). Then, we easily prove that UpdWit preserves (2) and (3) as well. Since (1) can be computed without K_s and that the game does not allow the adversary to add a member or to delete a non-member, all oracle calls in the game can be simulated without knowing K_s. So, the security is equivalent to forging $X \subseteq P$ and $x \in P$ together with W which is an incoherent proof for x with respect to X with K_p as a sole input. We can easily see that this implies breaking the strong RSA assumption (c.f. [7] for more details).

Quite importantly, it is necessary that all proofs of non-membership satisfy (3). Indeed, assuming that $x \notin X$ and we have some (a', d') such that $c^{a'} \equiv d'^x g (\bmod n)$, the adversary can compute (a, d) satisfying (3) by going through all X members and without requiring K_s. Then, he would obtain a relation $c^{a'-a} \equiv (d'/d)^x (\bmod n)$. If the proof (a', d') does not satisfy (3), then x does not divide $a' - a$. So, the adversary can invert $a' - a$ modulo x and obtain a relation $w^x \bmod n = c$ which is a proof of membership although x has been deleted. Security collapses. So, it is important to provide only proofs (a, d) such that (3) holds. A similar weakness was spotted in [10]. It was based on a proposal to compute (a, d) from $\prod_{y \in X} y \bmod \varphi(n)$ instead of $\prod_{y \in X} y$, which is quite a bad idea! The above procedure is safe. (Indeed, its computation does not require the secret key, so it is zero-knowledge.)

One drawback of LLX is that after deleting x from X we cannot create a proof of non-membership for x except by recomputing a from scratch, since (3) must be satisfied. Although it seems dynamic, it fails to provide the promised efficiency since we need X to compute a and, hence, only offers a partially dynamic universal accumulator. A similar drawback exists in the WitGen algorithm. Indeed, the authority willing to issue a proof of non-membership for a non-member which never needed such a proof has to go through the entire X structure.

3. WEAK DYNAMIC ACCUMULATORS FOR ARBITRARY DOMAINS

In this section, we define the notion of *weak* dynamic accumulator where elements can be dynamically added to the accumulator. It's called a weak dynamic accumulator because it only considers adding and not deleting elements, and that is all we need for our construction in Section 4. When compared to Dynamic Accumulators in Def. 11, WD accumulators do not require an AccVal to compute c from X.

Definition 5 (Weak Dynamic (WD) Accumulators). A weak dynamic accumulator WDAcc*, with a domain P, a set* $X \subseteq P$*, and values* $x \in X$ *to be accumulated, consists of the following algorithms.*

– *A setup probabilistic algorithm* $\mathrm{KeyGen}(1^k) \to (K_s, K_p)$*, where* K_s *is only used by the authority and* K_p *is public.*

– *An algorithm* $\mathrm{InitAccVal}(K_s, K_p) \to c$*, which computes an initial accumulator value* c *of the empty set, from the keys.*

– *An algorithm* $\mathrm{AddEle}(K_s, K_p, c, x) \to (c', \mathrm{extra})$*, which computes the accumulator* c' *for* $X + \{x\}$ *from the accumulator* c *for* X*. We must have* $x \notin X$ *and we say that* x *is inserted into* X*. The algorithm also returns some extra information* extra*, which might be needed for dynamic witness update.*

– *An algorithm* $\mathrm{UpdWit}(K_p, c, c', \mathrm{extra}, x, W, y) \to W'$ *to generate a proof* W' *for* y *in accumulator* c' *from a proof* W *for* y *in accumulator* c*, where* $\mathrm{AddEle}(K_s, K_p, c, x) \to (c', \mathrm{extra})$.

– *A predicate* $\mathrm{Verify}(K_p, c, x, W)$ *to check a proof.*

Note that all WD accumulators are dynamic, by definition. Hence, the constructions due to Li *et al.* or Camenisch and Lysyanskaya are both WD accumulators. Moreover, Def. 3, the correctness definitions, can be applied to WD accumulators considering $+$ as the only possibility for op. Furthermore, the security notion for a WD accumulator is a special case of a Chosen Element Attack scenario since there are no deletions.

Definition 6 (Security of a WD accumulator). *The security of the WD accumulator is defined in terms of a game, based on a security parameter* k*, played by a polynomially bounded adversary. Firstly,* KeyGen *is run and* K_p *is given to the adversary. Secondly, the adversary selects a polynomially bounded number* ℓ*. There are registers* K_s*,* X_i*, and* c_i*,* $i = 1, \ldots, \ell$*, for a secret key and to keep track of* ℓ *sets* X_i *and their accumulator values* c_i*. Initially, all* X_i*'s are empty and* c_i *is set to* $\mathrm{InitAccVal}(K_s, K_p)$*. The adversary can then call an* $\mathrm{AddEle}(K_s, K_p, c_i, .,.)$ *oracle for a selected* i *which updates* X_i *and* c_i *accordingly. It is not allowed to add an* x *to* X_i *when* x *is already in* X_i*. After making many oracle queries, the adversary ends by producing some* (i, x, W)*. The adversary wins if* W *is an incoherent proof for* x *with respect to* X_i *and accumulator* c_i*.*

We now describe a generic way of transforming a WD accumulator WDAcc_0 with domain P, set of elements x with some special form, to a WD accumulator with an arbitrary domain, *i.e.*, a finite subset of $\{0,1\}^*$. In this generic transformation, we will make use of a signature scheme. In the following description,

the algorithms, values, and predicates with index of 0, *e.g.*, KeyGen_0, refer to the corresponding items in WDAcc_0 defined as above, whereas values indexed by sig refer to those of a signature scheme.

We assume that the elements in P can be enumerated starting from $h = h_{\text{init}}$ and then iterating by means of an operation $h \leftarrow$next.element(h). The overall idea is that the value of the new accumulator consists of a pair $(c^0, \text{last.h})$, where c^0 is the accumulator value for WDAcc_0 and the last added last.h value. To add a new string x, we get a new h using next.element and we bind h to x using a signature on (x,h). A witness for x is a triplet (h,σ,w), where h is the bound value, σ is a valid signature, and w is a witness for h in WDAcc_0. When a new x is added, the witness for it is computed and returned as the extra information.

A generic transformation to obtain a WD accumulator with arbitrary domain:

Domain: S is a large enough subset of $\{0,1\}^*$.

KeyGen(1^k): run $\mathrm{KeyGen}_0(1^k)$ and obtain K_p^0 and K_s^0. Further, run $\mathrm{KeyGen}_{\text{sig}}(1^k)$ and obtain $(K_s^{\text{sig}}, K_p^{\text{sig}})$. Then $K_p = (K_p^0, K_p^{\text{sig}})$ and $K_s = (K_s^0, K_s^{\text{sig}})$.

InitAccVal(K_p,X) $\rightarrow$c=(c^0, h_{init}): where c^0 is the output of $\mathrm{InitAccVal}^0(K_s^0, K_p^0)$.

AddEle(K_s, K_p,(c^0, last.h),x): Let $h \leftarrow \text{next.element(last.h)}$ denote the next element in the list and call $\mathrm{AddEle}^0(K_s^0, K_p^0, c^0, h) \rightarrow (c^{0\prime}, \mathrm{extra}_0)$. Then let $\sigma \leftarrow \mathrm{sig}(K_s^{\text{sig}}, x, h)$. Return $((c^{0\prime}, h), (\sigma, \mathrm{extra}_0))$.

UpdWit(K_p,(c^0, last.h), ($c^{0\prime}$, next.h), (σ,extra), x, W, y): If $x = y$, then return $(\text{next.h}, \sigma, \text{extra})$. If $x \neq y$, extract h and w from $W = (h,\sigma,w)$ and then return $(h, \sigma, \mathrm{UpdWit}^0(K_p^0, c^0, c^{0\prime}, \text{extra}, \text{next.h}, w, h))$.

Verify(K_p, (c^0, last.h), x, (h,σ,w)): is true if and only if both $\mathrm{Verify}^{\text{sig}}(K_p^{\text{sig}}, (x,h), \sigma)$ and $\mathrm{Verify}^0(K_p^0, c^0, h, w)$ are true.

Very similar generic transformations exist in the literature, see for example [4], where it is suggested that the issuer of the accumulator would need to publish a mapping from S to P used along with a signature scheme. The advantage of our scheme compared to those approaches is that we do not require any mapping to be published: we just assume that the elements of P can be enumerated and that it is easy to move to the next element, given the previous one.

The following theorem states that starting from a correct and secure WD accumulator with domain P and using a secure signature scheme, we obtain a correct and secure WD accumulator with the arbitrary domain S with the above generic transformation.

Theorem 7. *Consider a correct and secure WD accumulator* WDAcc_0 *and a secure digital signature scheme* sig *(i.e., signatures are unforgeable under chosen message attacks). The resulting WD accumulator* WDAcc *of the above generic transformation is correct and secure.*

Proof. For simplicity, we show the proof for the case of deterministic algorithm, hence assume $\ell = 1$. The general case follows similarly.

The correctness follows immediately as both the signature's and the original accumulator's verification predicate are being verified. More precisely, we need to show that WDAcc is correct according to Def. 3. Since it is not a universal accumulator, all we need to show is that $W \vdash (x, X, c, b)$ implies $\mathrm{Verify}(K_p, c, x, W)$, for all probabilistic polynomial time adversaries, and for all possible choices of W, x, X, and c. Note that both AddEle and UpdWit call upon the respective algorithms in WDAcc_0. In particular, we have that $c = (c^0, h)$, for some h. Hence, $W \vdash (x, X, c, b)$ for WDAcc implies that

$W \vdash (h, X^0, c^0, b)$ for WDAcc_0. Now, since WDAcc_0 is a correct accumulator, $W \vdash (h, X^0, c^0, b)$ implies $\mathrm{Verify}(K_p, c^0, h, W)$, which in turn implies $\mathrm{Verify}(K_p, c, x, W)$.

Moreover, as the signature scheme is only binding elements of the two domains together, any incoherent witness with respect to the domain S of WDAcc produces an incoherent witness for a corresponding element in the domain P. More precisely, we can reduce an adversary $\mathcal{A}$ who can find an incoherent proof with respect to WDAcc to an adversary $\mathcal{B}$ who produces an incoherent proof with respect to WDAcc_0. We now outline this reduction.

According to the security game of Def. 6 for WDAcc, KeyGen is run and $K_p = (K_p^0, K_p^{\mathrm{sig}})$ is given to the adversary. Initially, X is empty and the accumulator is set to $\mathrm{InitAccVal}(K_s, K_p)$. In particular, the value of the accumulator c is (c^0, h_{init}), where c^0 is the output of $\mathrm{InitAccVal}^0(K_s^0, K_p^0)$. Then $\mathcal{A}$ can call the $\mathrm{AddEle}(K_s, K_p, c, .,.)$ oracle which updates X and c accordingly, but is not allowed to add an x to X when x is already in X. In particular, each AddEle query lets $h := \mathrm{next.element(last.h)}$, calls $\mathrm{AddEle}^0(K_s^0, K_p^0, c, h) \to (c^0, \mathrm{extra}_0)$, computes $\sigma := \mathrm{sig}(K_s^{\mathrm{sig}}, x, h)$, and returns $((c^0, h), (\sigma, \mathrm{extra}_0))$. After enough oracle queries, the adversary ends the game by producing some (x, W) which is an incoherent proof for x with respect to X and accumulator c, where $W = (x, \sigma_x, w)$.

Now let's look at the game played by $\mathcal{B}$. Firstly, KeyGen_0 is run and K_p^0 is given to the adversary. Initially, X^0 is empty and c^0 is set to $\mathrm{InitAccVal}^0(K_s^0, K_p^0)$. The adversary calls $\mathrm{AddEle}^0(K_s^0, K_p^0, c^0, .,.)$ oracles to update X^0 and c^0 accordingly, but is not allowed to add an x^0 to X^0 when x^0 is already in X^0. Once the adversary has made enough queries, she produces some (x^0, w). She wins if w is an incoherent proof for x^0 with respect to X^0 and accumulator c^0.

We are now going to use $\mathcal{A}$ to help $\mathcal{B}$ win his game. Upon receiving K_p^0, $\mathcal{B}$ runs $\mathrm{KeyGen}_{\mathrm{sig}}(1^k)$ and obtains $(K_s^{\mathrm{sig}}, K_p^{\mathrm{sig}})$. Then provides $\mathcal{A}$ with $K_p = (K_p^0, K_p^{\mathrm{sig}})$. For every $\mathrm{AddEle}(K_s, K_p, c, .,.)$ oracle query of $\mathcal{A}$, $\mathcal{B}$ does the following. He lets $h := \mathrm{next.element(last.h)}$, calls $\mathrm{AddEle}^0(K_s^0, K_p^0, c, h) \to (c^0, \mathrm{extra}_0)$, computes $\sigma := \mathrm{sig}(K_s^{\mathrm{sig}}, x, h)$, and returns $((c^0, h), \sigma, (\mathrm{extra}_0))$ to $\mathcal{A}$. Note that $\mathcal{B}$ posses K_s because he ran $\mathrm{KeyGen}_{\mathrm{sig}}(1^k)$ at the beginning of this reduction. Finally, $\mathcal{A}$ provides $\mathcal{B}$ with some (x, W) which is an incoherent proof for x with respect to X and accumulator c. Note that witnesses of WDAcc are of the form (h, σ, w), where $\sigma := \mathrm{sig}(K_s^{\mathrm{sig}}, x, h)$ and the accumulator c is of the form $((c^0, h), (\sigma, \mathrm{extra}_0))$. Hence, an incoherent witness (h, σ, w) of x in WDAcc implies an incoherent witness W of h in WDAcc_0. □

4. FULLY DYNAMIC UNIVERSAL ACCUMULATORS

In this section we formalize the notion of fully dynamic universal accumulator, then show how to construct some based on the LLX accumulator and a weak dynamic accumulator.

4.1. Definitions

We say that a dynamic universal accumulator is *fully dynamic* if the following conditions are satisfied:

1. We can always create a non-membership proof for a new x (*i.e.*, a value x occurring for the first time or a newly deleted x) without using X.

2. We can create a proof of membership without using X for a newly added x.

That is, we can create proofs for newly occurring values x (*e.g.*, non-members) with an algorithm which does not depend on the cardinality of X. To make this change possible, we introduce a new operation dec in UpdEle to "declare" new non-members, and we make UpdWit run with $x = y$ without any prior witness W for operations + and dec. This UpdEle can be used to compute the initial proof of non-membership for x. However, it requires to update the accumulator value *c*. More formally, we define a fully dynamic universal accumulator as follows.

Definition 8 (Fully Dynamic Universal (FDU) Accumulator). A fully dynamic universal accumulator FDUAcc*, with a domain P, a set* $X \subseteq P$*, and values* $x \in X$ *to be accumulated, consists of the following algorithms.*

– A setup probabilistic algorithm $\mathrm{KeyGen}(1^k) \to (K_s, K_p)$*, where* K_s *is only used by the authority and* K_p *is public.*

– An algorithm $\mathrm{AccVal}(K_s, K_p, X) \to c$*, which computes an accumulator value* c *of the set X, from the keys.*

– An algorithm $\mathrm{UpdEle}(K_s, K_p, c, \mathrm{op}, x) \to (c', \mathrm{extra})$*, where* $\mathrm{op} = +$*,* $\mathrm{op} = -$*, or* $\mathrm{op} = \mathrm{dec}$*, which computes the accumulator* c' *from an accumulator* c*. When* $\mathrm{op} = +$*, we must have* $x \notin X$ *and we say that* x *is inserted into X. When* $\mathrm{op} = -$*, we must have* $x \in X$ *and we say that* x *is deleted from X. When* $\mathrm{op} = \mathrm{dec}$*, we must have* $x \notin X$ *and we say that* x *is declared as a non-member of X. The algorithm also returns some extra information* extra*. For* $\mathrm{op} = \mathrm{dec}$*, the algorithm also returns some new proof of non-membership for* x *(if not already in* extra*).*

– An algorithm $\mathrm{WitGen}(K_s, c, X, x) \to W$ *to generate a proof of membership or non-membership for the value* x *with respect to accumulator* c *of X.*

– An algorithm $\mathrm{UpdWit}(K_p, \mathrm{extra}, W, y) \to W'$ *to generate a proof* W' *for* y *in accumulator after* UpdEle *returned* extra *from a previous proof W.*

– A predicate $\mathrm{IsMem}(W)$ *telling whether* W *is a proof of membership (true case) or a proof of non-membership (false case).*

– A predicate $\mathrm{Verify}(K_p, c, x, W)$ *to check a proof.*

Note that UpdWit no longer requires that y is different from the element involved in the last UpdEle call.

The correctness notion is similar to that of WD accumulators with the obvious change that the witnesses are not only produced by UpdWit, but also by WitGen and that witnesses are either for membership or non-membership proofs. This change was foreseen in our general correctness notion in Def. 3.

Next, we detail a variant of the Chosen Element Attack scenario corresponding to FDU accumulators, in which the adversary is allowed to declare new elements as well as add or delete.

Definition 9 (Extended Chosen Element Attack (ECEA) Model). *The security of an FDU accumulator is defined in terms of a game, based on a security parameter k, played by a polynomially bounded adversary. Firstly,* KeyGen *is run and* K_p *is given to the adversary. Secondly, the adversary selects a polynomially bounded number* ℓ*. There are registers* K_s*,* X_i*, and* c_i*,* $i = 1, \ldots, \ell$*, for a secret key and to keep track of* ℓ *sets* X_i *and their accumulator values* c_i*. Initially, all* X_i*'s are empty and* c_i *is set to* $\mathrm{AccVal}(K_s, K_p, X_i)$*. The adversary can then call an* $\mathrm{UpdEle}(K_s, K_p, c_i, .,.)$ *oracle for a selected* i

which updates X_i *and* c_i *accordingly. It is not allowed to add an* x *to* X_i *when* x *is already in* X_i*, nor is it allowed to delete* x *from* X_i *when* x *is not in* X_i*. Moreover, the adversary is not allowed to declare an element which has already been declared, i.e., is either a member or a non-member. The adversary can also call a* $\mathrm{WitGen}(K_s, c_i, X_i, .)$ *oracle for a selected* i *and an* $\mathrm{AccVal}(K_s, K_p, .)$ *oracle which do not update an* X_i *of* c_i*. After making many oracle queries, the adversary ends by producing some* (i, x, W)*. The adversary wins if* W_X *is an incoherent proof for* x *with respect to* X_i *and accumulator* c_i*.*

4.2. Instantiating an FDU Accumulator based on the Strong RSA Assumption

Below we construct a fully dynamic universal accumulator based on a variant of the LLX accumulator. The main idea relies on providing a two-layer accumulator. The lower layer c_2 is a WD accumulator where we accumulate all declared elements. *I.e.*, all values which have ever be used, whether they are member or not. There is no withdrawal in this layer. The upper layer c_1 is a fully dynamic accumulator which can only treat elements of the lower layer. Essentially, (a, d, h, w) is a proof of non-membership for x in accumulator (c_1, c_2), if $c_1^a \equiv d^x h \pmod{n_1}$ and w is a proof of membership for h in accumulator c_2, *i.e.*, $w^h \bmod n_2 = c_2$. To create a proof of non-membership for a newly deleted member or a new comer who is not a member, we only have to pick a random $a \in \mathrm{Z}_x$ and $d \in \mathrm{Z}_n^*$ and compute the corresponding h to add in the second accumulator. That is, the WD accumulator is only used to validate new h values which are needed to create new proofs. Since proofs also have a part in the PDU accumulator, it is not necessary to delete h from the WD accumulator.

Equation (3) is now replaced by

$$a = a_0 \frac{\prod_{y \in X_0} y}{\prod_{y \in X} y} \bmod x \quad \textit{and} \quad d = \left(c_1^a h^{-1}\right)^{\frac{1}{x}} \bmod n_1, \tag{4}$$

where a_0, respectively X_0, is the initial value for a, respectively X, and a_0 is chosen at random.

Our accumulator is defined as follows. The value of c is defined by $c = (c_1, c_2)$ corresponding to two accumulators c_1 and c_2. It will be convenient in the security game to define two sets X_1, defined as before, and X_2, a finite set of elements, corresponding to this value c.

The set X_1 is updated by UpdEle and is accumulated in the value c_1, whereas the set X_2 is accumulated in c_2 by means of a WD accumulator with arbitrary domain. Hence, the value of c is deterministically defined by c_1 and c_2, where

$$c_1 = g^{\prod_{x \in X_1} x} \bmod n_1.$$

The value of c will corresponds to $X = X_1$. So, there are many c's corresponding to the same X depending on X_2.

In the following description, the algorithms, values, and predicates with index of 1, *e.g.*, KeyGen_1, refer to the corresponding items in the PDU accumulator of Li *et al.* [7], whereas the algorithms, values, and

predicates with index of 2, *e.g.*, Verify_2, refer to those of a WD accumulator with arbitrary domain as defined in Section 3.

A Concrete FDU accumulator:

Domain: P is a large enough set of odd prime numbers, *e.g.*, all odd numbers up to a given bound B.

KeyGen(1^k): run $\mathrm{KeyGen}_1(1^k)$ and obtain $K_p^1 = (n_1, g_1)$ and $K_s^1 = r_1$. Further, run $\mathrm{KeyGen}_2(1^k)$ and obtain (K_s^2, K_p^2). Then $K_p = (K_p^1, K_p^2)$ and $K_s = (K_s^1, K_s^2)$.

AccVal(K_p,X): compute

$$c_1 = g^{\prod_{x\in X} x} \bmod n_1$$

and return $c = (c_1, \mathrm{InitAccVal}_2(K_s^2))$. Note that K_s is not used.

UpdEle(K_s, Kp, c, op, x): if $\mathrm{op} = +$, we have $c_1' = c_1^x \bmod n_1$ and $c_2' = c_2$ (K_s^1 is not required). Set extra = $(c,+,x,c')$. If op = $-$, we set $c_1' = c_1^{\frac{1}{x}} \bmod n_1$ (K_s^1 is required) and proceed like for op = dec. If op = dec, we pick $a \in \mathrm{Z}_x$ and $d \in \mathrm{Z}_{n_1}^*$ at random, and compute $h = c_1^a d^{-x} \bmod n_1$, then we set $W = (\mathrm{nonmem}, (a, d, h, c_2))$. We set $(c_2', \mathrm{extra}_2) = \mathrm{AddEle}_2(K_s^2, K_p^2, c_2, h)$. The extra information is then set to $\mathrm{extra} = (c, \mathrm{op}, x, c', W, \mathrm{extra}_2)$.

WitGen(K_s, c, X, x): if $x \in X$, then $W = (\mathrm{mem}, w)$ with $w = c_1^{\frac{1}{x}} \bmod n_1$. This requires K_s, but not X. It can also be computed without K_s, but using X by observing that

$$w = c_1^{\prod_{y\in X-\{x\}} y} \bmod n_1.$$

If $x \notin X$, then $W = (\mathrm{nonmem}, (a, d))$ as in the [LLX] accumulator by using X.

UpdWit(K_p, extra, W,y): with $\mathrm{extra} = (c, \mathrm{op}, x, c', e, \mathrm{extra}_2)$, there a several cases to update the proof for y in X after adding or deleting x for $x \neq y$:

- W of the form (mem, w) and x just added ($\mathrm{op} = +$): set $W' = (\mathrm{mem}, w')$ with $w' = w^x \bmod n_1$.
- W of the form (mem, w) and x just deleted ($\mathrm{op} = -$): set $W' = (\mathrm{mem}, w')$ with $w' = w^z c'^{\frac{1-xz}{y}} \bmod n_1$ and $z = \frac{1}{x} \bmod y$.
- W of the form (mem, w) and x just declared ($\mathrm{op} = \mathrm{dec}$): $W' = W$.
- W of the form (nonmem, a, d) and x just added ($\mathrm{op} = +$): set $W' = (\mathrm{nonmem}, [a', d']_y)$ with $a' = az$, $d' = dc^{-a\frac{1-xz}{y}} \bmod n_1$ and $z = \frac{1}{x} \bmod y$.
- W of the form (nonmem, a, d) and x just deleted ($\mathrm{op} = -$): set $W' = (\mathrm{nonmem}, [a', d]_y)$ with $a' = ax$.
- W of the form (nonmem, a, d) and x just declared ($\mathrm{op} = \mathrm{dec}$): $W' = W$.
- W of the form $(\mathrm{nonmem}, a, d, h, w)$ and x just added ($\mathrm{op} = +$): set $W' = (\mathrm{nonmem}, [a', d']_y, h, w)$ with $a' = az$, $d' = dc^{-a\frac{1-xz}{y}} \bmod n_1$ and $z = \frac{1}{x} \bmod y$.

– W of the form $(\text{nonmem}, a, d, h, w)$ and x just deleted ($\text{op} = -$): set $W' = (\text{nonmem}, [a', d]_y, h, w')$ with $a' = ax$ and $w' = \text{UpdWit}_2(K_p^2, c_2, c_2', \text{extra}_2, w, h)$.

– W of the form $(\text{nonmem}, a, d, h, w)$ and x just declared ($\text{op} = \text{dec}$): $w' = \text{UpdWit}_2(K_p^2, c_2, c_2', \text{extra}_2, w, h)$.

For $x = y$, there are two cases:

– x just added ($\text{op} = +$): set $W' = (\text{mem}, w)$ with $w = c_1$ from $c = (c_1, c_2)$.

– x just deleted ($\text{op} = -$) or just declared ($\text{op} = \text{dec}$): set $W' = e$.

IsMem(W): is true if and only if W is of form $(\text{mem},.)$.

Verify(*Kp, c,x,W*): if W is of the form (mem, w), it is true if and only if $c_1 = w^x \bmod n_1$. If W is of form (nonmem, a, d), it is true if and only if $c^a \equiv d^x g (\bmod n_1)$. If W is of the form $(\text{nonmem}, a, d, h, w)$, it is true if and only if $c_1^a \equiv d^x h (\bmod n_1)$ and $\text{Verify}_2(K_p^2, c_2, w, h)$ holds.

We now prove the correctness and security of our FDU accumulator scheme based on the Strong RSA assumption.

Theorem 10. *If the Strong RSA Assumption holds, the aforementioned FDU accumulator is correct and secure under the ECEA model.*

Proof. The correctness follows immediately since both PDU and WDU considered as building blocks of our FDU are correct accumulators according to Def. 3.

All oracle calls in the security game preserve the relations described in the AccVal and WitGen algorithms. Furthermore, since the game does not allow the adversary to add a member or to delete a non-member, all oracle calls in the game can be simulated without knowing K_s^1. Hence, the security is equivalent to forging $X \subseteq P$ and $x \in P$ together with W which is an incoherent proof for x with respect to X and K_p as the only input. Hence, an incoherent witness implies breaking the strong RSA assumption. In other words, computing an incoherent witness for $x \in X$ implies an incoherent witness for the PDU accumulator of Li *et al.* [7] and computing an incoherent witness for $x \notin X$ implies that, given n_1 and a random c_1 drawn from $\mathsf{Z}_{n_1}^*$, the adversary has found $w \in \mathsf{Z}_{n_1}^*$ and $x > 1$ such that $c_1 = w^x \bmod n_1$.

More precisely, we can reduce an adversary $\mathcal{A}$ who produces incoherent proofs in the aforementioned FDU accumulator to an adversary $\mathcal{B}$ who produces incoherent proofs in either the WD accumulator with arbitrary domain as defined in Section 3 or the PDU accumulator of Li *et al.* [7], which, in turn, implies an adversary who breaks the Strong RSA Assumption. The reduction is detailed below.

According to Def. 9, the security game starts with running KeyGen and giving K_p to the adversary. That is, both KeyGen_1 and KeyGen_2 are run to obtain $K_p^1 = (n_1, g_1), K_s^1 = r_1$, and (K_s^2, K_p^2). Then, $K_p = (K_p^1, K_p^2)$ is given to the adversary. Initially, X is empty and c is set to $\text{AccVal}(K_p, X)$. That is, $c_1 = g^{\prod_{x \in X} x} \bmod n_1$ is computed and $c = (c_1, \text{InitAccVal}_2(K_s^2))$ is returned. Note that K_s is not used in this computation.

Then, the adversary calls $\text{UpdEle}(K_s, K_p, c, .,.)$ oracle queries to update X and c accordingly. She is not allowed to add an x to X when x is already in X, nor is she allowed to delete x from X when x is not in *X*. Moreover, the adversary is not allowed to declare an element which has already been declared, *i.e.*, is either a member or a non-member. When an element x is being added, we have $c_1' = c_1^x \bmod n_1$, *i.e.*,

using the PDU accumulator of Li *et al.* [7], and $c_2' = c_2$, *i.e.*, in the WDU accumulator. If x is being declared, random $a \in \mathrm{Z}_x$ and $d \in \mathrm{Z}_{n_1}^*$ are picked to compute $h = c_1^a d^{-x} \bmod n_1$, again as in the PDU accumulator of Li *et al.* [7], to obtain $W = (\text{nonmem}, (a, d, h, c_2))$. If the element x is being deleted, we have $c_1' = c_1^{\frac{1}{x}} \bmod n_1$ and proceed like the declaration process. Note that K_s^1 is not required in any of these steps.

The adversary can also call a $\mathrm{WitGen}(K_s, c, X, .)$ oracle query. If $x \in X$, then $W = (\text{mem}, w)$ with $w = c_1^{\prod_{y \in X - \{x\}} y} \bmod n_1$, which can be computed without K_s, but using X. If $x \notin X$, then $W = (\text{nonmem}, (a, d))$ as in the PDU accumulator of Li *et al.* [7]. Note that, again, K_s is not required in any of these steps.

Once the adversary has made enough oracle queries, she ends the game by producing some (x, W) that is an incoherent proof for x with respect to X and accumulator $c = (c_1, c_2)$. The witness W is of the form (mem, w), (nonmem, a, d), or $(\text{nonmem}, a, d, h, w)$. We are going to consider each case separately.

– If W, the incoherent proof, is of the form (mem, w), then, by definition of the verification algorithm Verify, we must have that $c_1 = w^x \bmod n_1$, which directly breaks the Strong RSA assumption.

– In order for an incoherent witness W of form (nonmem, a, d) to pass the verification step, we must have that $c^a \equiv d^x g (\bmod n_1)$. This translates to an incoherent witness for the PDU accumulator of Li *et al.* [7].

– If the incoherent witness W is of the form $(\text{nonmem}, a, d, h, w)$. Then, both $c_1^a \equiv d^x h (\bmod n_1)$ and $\mathrm{Verify}_2(K_p^2, c_2, w, h)$ must hold for it to pass the verification step. This translates to either an incoherent witness for the PDU accumulator of Li *et al.* [7] or an incoherent proof for the WD accumulator of Section 3.

Hence, an adversary who can find incoherent witnesses for our FDU accumulator is capable of producing incoherent proofs for the WD accumulator with arbitrary domain as defined in Section 3 or the PDU accumulator of Li *et al.* [7], both of which are based on the Strong RSA assumption.

□

Note that setting $g = h$ reduces the structure of our non-membership proofs to that of Li *et al.* [7].

We point out that the efficacy of our proof structure allows the authority to perform efficient batch updates (with the secret key) for a given value x. The authority first checks to see whether x is a member of the accumulated set or not. If a member, then using the same procedure as in the scheme of Li *et al.* the authority can efficiently update the witness. This is not incompatible with the impossibility of batch update without the secret key [3]. However, the scheme of Li *et al.* did not offer such a mechanism for a non-member element. In our scheme, we can create a new non-membership proof deploying the mechanism for declaration.

5. CONCLUSIONS AND FUTURE WORK

We constructed the first *fully dynamic* universal accumulator, based on the Strong RSA assumption, by providing a new proof structure for the non-membership witnesses. Moreover, this new structure of non-membership proofs allows our scheme to be the first of its kind to offer an *efficient batch update* mechanism to the authority, for both members and non-members. We obtained our fully dynamic universal accumulator by means of deploying a weak dynamic accumulator with arbitrary domain, which we showed how to obtain from a weak dynamic accumulator with a domain of certain form.

ACKNOWLEDGEMENTS

We would like to thank Nokia for initiating this work, as well as Rafik Chaabouni who contributed.

A. EXTRA DEFINITIONS

Definition 11 (Dynamic Accumulators). A dynamic accumulator DAcc, *with a domain* P, *a set* $X \subseteq P$, *and values* $x \in X$ *to be accumulated, consists of the following algorithms.*

– *A setup probabilistic algorithm* $\mathrm{KeyGen}(1^k) \to (K_s, K_p)$, *where* K_s *is only used by the authority and* K_p *is public.*

– *An algorithm* $\mathrm{AccVal}(K_s, K_p, X) \to c$, *which computes an accumulator value* c.

– *An algorithm* $\mathrm{UpdEle}(K_s, K_p, c, \mathrm{op}, x) \to (c', \mathrm{extra})$, *where* $\mathrm{op} = +$ *or* $\mathrm{op} = -$, *which computes the accumulator* c' *for* $X\mathrm{op}\{x\}$ *from the accumulator* c *for* X. *When* $\mathrm{op} = +$, *we must have* $x \notin X$ *and we say that* x *is inserted into* X. *When* $\mathrm{op} = -$, *we must have* $x \in X$ *and we say that* x *is deleted from* X. *The algorithm also returns some extra information* extra, *which might be needed for dynamic witness update.*

– *An algorithm* $\mathrm{WitGen}(K_s, c, X, x) \to W$ *to generate a proof of membership for the value* x *with respect to accumulator* c *of* X.

– *An algorithm* $\mathrm{UpdWit}(K_p, c, c', \mathrm{extra}, \mathrm{op}, x, W, y) \to W'$ *to generate a proof* W' *for* y *in accumulator* c' *from a proof* W *for* y *in accumulator* c, *where* $\mathrm{UpdEle}(K_s, K_p, c, \mathrm{op}, x) \to (c', \mathrm{extra})$. *It must be the case that* $x \neq y$.

– *A predicate* $\mathrm{Verify}(K_p, c, x, W)$ *to check a proof.*

Definition 12 (Universal Accumulators). A universal accumulator *scheme* UAcc, *with a domain* P *a set* $X \subseteq P$, *and values* $x \in X$ *to be accumulated, consists of the following algorithms.*

– *A setup probabilistic algorithm* $\mathrm{KeyGen}(1^k) \to (K_s, K_p)$, *where* K_s *is only used by the authority and* K_p *is public.*

– *An algorithm* $\mathrm{AccVal}(K_s, K_p, X) \to c$, *which computes an accumulator value* c.

– *An algorithm* $\mathrm{MemWitGen}(K_s, c, X, x) \to W$ *to generate a proof of membership for* $x \in X$.

– *An algorithm* $\mathrm{NonMemWitGen}(K_s, c, X, x) \to W$ *to generate a proof of non-membership for* $x \in P \setminus X$.

– *A predicate* $\mathrm{IsMem}(W)$ *telling whether* W *is a proof of membership (true case) or a proof of non-membership (false case).*

– *A predicate* $\mathrm{Verify}(K_p, c, x, W)$ *to check a proof.*

REFERENCES

1. Josh Cohen Benaloh and Michael De Mare, *One-way accumulators: A decentralized alternative to digital sinatures (extended abstract)*, in EUROCRYPT 1993, pp. 274–285, 1993.
2. Niko Barić and Birgit Pfitzmann, *Collision-free accumulators and fail-stop signature schemes without trees*, in Proceedings of the 16th annual international conference on Theory and application of cryptographic techniques, EUROCRYPT 1997, pp. 480–494, Berlin, Heidelberg, 1997. Springer-Verlag.
3. Philippe Camacho and Alejandro Hevia, *On the Impossibility of Batch Update for Cryptographic Accumulators*, LATINCRYPT 2010, volume 6212 of Lecture Notes in Computer Science, pp. 178–188, 2010.

4. Jan Camenisch, Markulf Kohlweiss, and Claudio Soriente, *An accumulator based on bilinear maps and efficient revocation for anonymous credentials* Stanislaw Jarecki and Gene Tsudik, editors, PKC 2009, volume 5443 of Lecture Notes in Computer Science, pp. 481–500, Springer, 2009.
5. Jan Camenisch and Anna Lysyanskaya, *Dynamic accumulators and application to efficient revocation of anonymous credentials*, in Moti Yung, editor, CRYPTO 2002, volume 2442 of Lecture Notes in Computer Science, pp. 61–76. Springer, 2002.
6. Nelly Fazio and Antonio Nicolosi, *Cryptographic Accumulators: Definitions, constructions and applications*, Manuscript, 2003.
7. Jiangtao Li and Ninghui Li and Rui Xue, *Universal Accumulators with efficient nonmembership proofs*, Jonathan Katz and Moti Yung, editors, ACNS 2007, volume 4521 of Lecture notes in Computer Science, pp. 253–269. Springer, 2007.
8. Lan Nguyen, *Accumulators from bilinear pairings and applications*, Alfred Menezes, editor, CT-RSA 2005, volume 3376 of Lecture Notes in Computer Science, pp. 275–292. Springer, 2005.
9. Kaisa Nyberg, *Fast Accumulated Hashing*, FSE 1996, volume 1039 of Lecture Notes in Computer Science, pages 83-87. Springer, 1996.
10. Kun Peng and Feng Bao, *Vulnerability of a non-membership proof scheme*, SECRYPT 2010, pages 419-422. SciTePress, 2010.
11. Peishun Wang, Huaxiong Wang, and Josef Pieprzyk, *A new dynamic accumulator for batch updates*, Sihan Qing, Hideki Imai, and GuilinWang, editors, ICICS 2007, volume 4861 of Lecture Notes in Computer Science, pp. 98–112. Springer, 2007.
12. Peishun Wang, Huaxiong Wang, and Josef Pieprzyk, *Improvement of a dynamic accumulator at icics 07 and its application in multi-user keyword-based retrieval on encrypted data*, in APSCC 2008, pp. 1381–1386. IEEE, 2008.

Received July 15, 2013

THE PUBLISHING HOUSE
OF THE ROMANIAN ACADEMY

PROCEEDINGS OF THE ROMANIAN ACADEMY, Series A,
Volume 14, Special Issue 2013, pp. 286–294

THE THREAT LANDSCAPE OF PKI: SYSTEM AND CRYPTOGRAPHIC SECURITY OF X.509, ALGORITHMS, AND THEIR IMPLEMENTATIONS

Blaine HEIN

NATO Communications and Information Agency
E-mail: blaine.hein@ncia.nato.int

With the unavoidable reliance on public key cryptography in modern communications and information systems (CIS) it is critical to maintain visibility into the threat landscape which can adversely impact the trust in public key infrastructure (PKI) implementations. This knowledge is useful as an input to a risk analysis process to determine whether current PKI practices are sufficient, and to determine when to migrate to new algorithms, key lengths, or procedures. This paper provides a discussion of the main attacks against PKI systems, both system and cryptographic in origin. This paper suggests appropriate methods to strengthen PKI systems against these attacks and provides references for additional reading on these attacks.

Key words: Asymmetric Cryptography, Diginotar, Flame, Malware, MITM, Public Key Infrastructure, Social Engineering, Stuxnet.

1. INTRODUCTION

In the last year we have seen a substantial increase in discussions touching on the malicious use of public key cryptography and on some high profile failures of PKI systems. This paper provides analysis and discussion on many of these events, and groups the threats against PKI in two general categories: system security and cryptographic security.

The topics covered under system security might just as easily be seen in an analysis of the threats against any CIS whether the system relied on cryptography or not. Many of these types of attacks have been occurring since before PKI was available as a commercial product. Social engineering, Trojan horses, and malware are terms which have been fully entrenched into our vocabulary.

The information covered under the topic of cryptographic security has a mathematical or algorithmic basis. Over the years the debates on cryptographic security have concentrated on when an algorithm would be broken, and whether it would be broken due to an increase in processing power, or by a new advancement in the field of mathematics.

This paper is not intended to provide a full breakdown of virus implementation, nor is it designed to turn the reader into a crypto mathematician. For those with aspirations in this direction, I have provided a detailed list of references for further reading. Nor will it go into details on the types of mischief an attacker can get into once he has breached a system. This is left to the imagination (or nightmares) of the reader.

This paper will provide recommendations of security mechanisms which will make the exploitation more difficult.

2. SYSTEM SECURITY

Included under the topic of system security, are a variety of topics ranging from the social engineering of a single certificate, all the way to the loss of control of multiple PKI infrastructures. It is important to remember that trust is a critical concept with PKI, and if that trust is convincingly undermined, the loss of trust can exceed the scope of the actual incident.

2.1. Diginotar

The Diginotar attack was a major news event in 2011 made public by the use of a rogue google.com certificate in a large man in the middle (MITM) attack targeting Iranian Internet users. For a complete report on the breach of the Certificate Authorities (CA) the final report on the Black Tulip incident [1] can be downloaded from Fox-IT in The Netherlands. This is an excellent example of System security failures. The following paragraphs provide a high level summary of the attack, and some recommended countermeasures to consider.

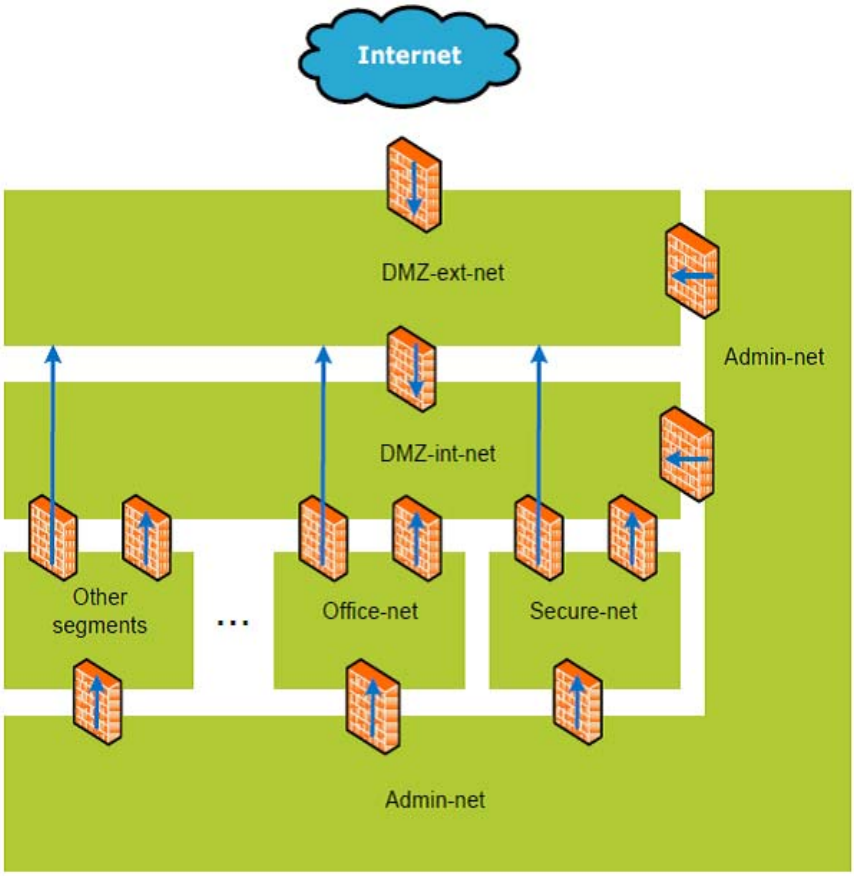


Fig. 1 – Diginotar Network Architecture [1, p. 18].

2.1a. Attack Progression

Reconnaissance and Scanning. Details are very limited within the Fox-IT report; however, it appears that the reconnaissance period for this attack was completed by 17 June 2011. From this point in time, the attack progressed steadily towards the end goal over the course of 3 weeks. Evidence in the report indicates that the attackers likely exploited a known vulnerability on an outdated software application and then used a user name and password discovered on the external web server to attempt connections into a database server.

Further evidence to support an undocumented reconnaissance period was found in the malicious code left on the systems by the attackers. These applications included hard coded IP addresses for internal and external systems, indicating that they were purpose modified for the Diginotar network [1, p. 48].

Exploiting Systems. First points of entry into the network were web servers located in the DMS [1, p. 23]. Next, database and other systems within the office network were compromised over the following 12 days. Tools to allow for the tunnelling of remote desktop protocol (RDP) sessions across non-standard ports (to allow for traversal of the firewalls) were discovered on systems in both the de-militarized zone (DMZ) and the office network.

The third step was the compromise of the secure internal network which included the Certificate Authority servers. Once access was gained to the CA servers, the attackers commenced with attempts to generate rogue SSL certificates. The first successful rogue certificate is thought to have been generated 8 days later.

Keeping Access and Covering the Tracks. Scripts were installed onto the web servers in the DMS to perform the function of a file system for uploading and downloading files.

Auditing subsystems were stopped in several cases to prevent the collection of forensic evidence while the attacker worked on the following steps of the attacks. Over 500 rogue certificates were issued. Some of the rogue certificates were issued with identical distinguished names (but possibly unique key pairs). Tools capable of exporting software certificates and private keys from the servers were found on multiple systems.

CA Database files were manipulated to hide the issuance of some certificates. As these files have integrity protection added by the CA, the easiest approach might have been to restore the database from a previous backup after issuing the certificates. This has the effect to return the CA to the state it was in prior to the issuance of those certificates, and as the integrity mechanisms on the backup file did not need to be tampered with, the files would appear to be intact. The biggest indications of attack are the lack of PKI audit logs during a period which shows attacks against the system from other audit mechanisms.

Certificate Authority implementation flaws exploited. Physical security at the Diginotar site was quite significant with the requirement to pass through multiple security zones and present visual, cryptographic, and biometric authentication data in order to gain physical access to the certificate authorities.

In the end, none of these security mechanisms came to bear to prevent the network based attack. The report [1] does not go into specific details regarding the activities performed on the Certificate Authorities to generate rogue certificates.

The flaws discussed below do not necessarily represent deficiencies within the PKI products. Instead, they reflect choices for the implementation of security mechanisms or configuration options.

Based on the level of details provided within the report, the following conclusions have been drawn.

2.1b. Countermeasures

Transitive Trust. Appendix IX of the report [1, p. 97] shows that five of the eight CAs included a common trust relationship allowing common administrative access between these systems. As soon as administrative privilege escalation was achieved on any system within this domain, all systems with this transitive trust relationship were effectively compromised. Removing transitive trust relationships from between the independent CAs would have added additional work load to gain access to the separate CAs.

Ubiquitous Strong Authentication. Authentication is composed of up to three distinct factors: What you (and only you) know, what you (and only you) have, and what you (and only you) are. Within this context, strong authentication implements at least two of the three factors. The end to end enforcement of strong authentication is critical for the infrastructure components of PKI due to the wide reaching impact of system compromise or suspected system compromise.

The [Black Tulip] report indicates that a smart card is required to log in directly to the CA software. However, there is no reference to strong authentication for the Diginotar Subscription Registration Production Interface (DARPI) application which performs part of the registration authority (RA) functionality.

Additionally, the mandatory use of strong authentication for all system login would decrease the attack surface. Finally, to mitigate against keystroke logging, implementing a smart card reader with an integrated keypad would prevent the smart card PIN from entering into the general system memory.

Separation of roles. PKI implementations normally include the concept of least privilege. Industry best practice such as the Certificate Issuing and Management Components (CIMC) protection profile (PP) requires that there are at least 3 orthogonal roles required to operate the certificate authority. In addition to the CA software, the Networked Hardware Security modules (nCipher netHSM) also implement separate strong authentication.

The automated issuance of CRL's requires that the netHSM be activated during the functioning period of the Certificate Authority. There should, however, be a separate strong authentication required between the Registration Authority (RA) and the CA to mitigate the risk that the active netHSM could be used to generate rogue certificates.

Use of In-house applications. Three in house applications were used to perform the registration authority and card personalization functions. No information is provided in this report as to the independent verification and validation processes that these applications may have been subjected to, or the security mechanisms that they implemented.

Protection of Audit Data. Storage of audit data on the same server as the PKI provides an easy path to covering the trail of the attacker. Deletion of the audit data, combined with restoring a previous backup of the CA database is a fast way to hide the existence of rogue certificates.

If the attacker had more completely erased his actions within the CA systems, (and not generated over 300 certificates which remained in the database) suspicion might have been limited to the compromise of a single CA. The attack could possibly have been repeated by substituting a new google certificate issued by another CA.

Pushing audit data to a separate system with further separation of roles for access control would add an additional layer of security in addition to providing log centralisation and normalisation functionality.

White listing Revocation Data. Certificate Revocation Lists (CRL)s implement a mechanism for blacklisting certificates when they are known to be compromised. OCSP provides an option to implement either blacklisting or white listing. To implement white listing, the CA must provide an explicit status response for all issued certificates. Any certificate which does not match these existing responses is deemed to be compromised (or fraudulent).

2.2. Social Engineering of Certificates

It has already been more than a decade since VeriSign fell victim to a social engineering attack in 2001, and issued two fraudulent code signing certificates with distinguished names linked to Microsoft Corporation. At the time, there was substantial discussion regarding how to remove these certificates and on the ability of Microsoft software to check revocation lists [2]. The Microsoft knowledge base article [3] outlines the mechanisms that were put in place to counter this threat. These certificates still exist in the Microsoft untrusted certificates store. Today, the largest issue with SSL implementations is the difficulty in determining whether the issuer of the SSL certificate is authoritative to issue that certificate. This conundrum exists because there is no correlation between DNS names and PKI Certificate Authorities, and there will likely never be one. To minimize this threat, minimize the number of root certificates trusted by your critical infrastructure components. Ensure this process is well tested before roll out to avoid a self-denial of service.

2.3. Theft of Certificates and Private Keys

Software implementations of cryptographic modules are vulnerable to extraction of private keys from their hosts. This is even more of a risk if these keys are stored on general purpose computing environments running commercial operating systems.

Readily available tools such as Mimikatz [4] have the ability to export keys from Microsoft security stores, even if the keys are marked as non-exportable. With these tools, the compromise of a server must be equated with the compromise of all private keys stored in that system.

Even when the cryptographic module implements a separate password to encrypt the private key, it is still vulnerable to brute force offline password guessing.

Hardware cryptographic modules, separation of roles, and two factor authentications must all be provided to protect critical certificates and keys. Code signing certificates are often overlooked in this scenario.

2.4. Compromise of Registration Authority

The attack against Comodo in the spring of 2011 appears to be the precursor to the Diginotar attack. In this case, the breach was to a registration authority (RA) which allowed the attacker to request and receive nine fraudulent certificates. Microsoft has published instructions to mitigate against these certificates at [5] and an online article [6] provides more background.

The same set of security mechanisms proposed to mitigate the Diginotar attack would likely have hampered the success in this attack as well.

2.5. Supply Chain Security

Security is only as strong as its weakest link. Sometimes even that statement is optimistic. The breach of servers at RSA through an advanced persistent threat (APT) [7] attack resulted in the compromise of a large number of SecurID tokens. However, RSA was not the end target in this attack. The compromised information was used to attack 3 US defence contractors [8]. While SecurID is not a PKI based security mechanism, what would the result be if you outsourced your PKI to a vendor who is subsequently breached? Ensure that your information security management system includes linkages to all aspects of your supply chain. What security mechanisms have your consultants and contractors implemented for IT equipment they connect to your network? Will the compromise of one of your suppliers or business partners result in a breach to your network because of trust relationships?

2.6. Unintentional Consequences of Security Implementations

2.6a. SSL Proxy Implementations

SSL proxies are appliances configured to decrypt SSL connections initiating from within a corporate environment outbound to commercial web services. They are implemented generally to support the implementation of malicious code scanning and data loss protection at the boundary of the corporate network.

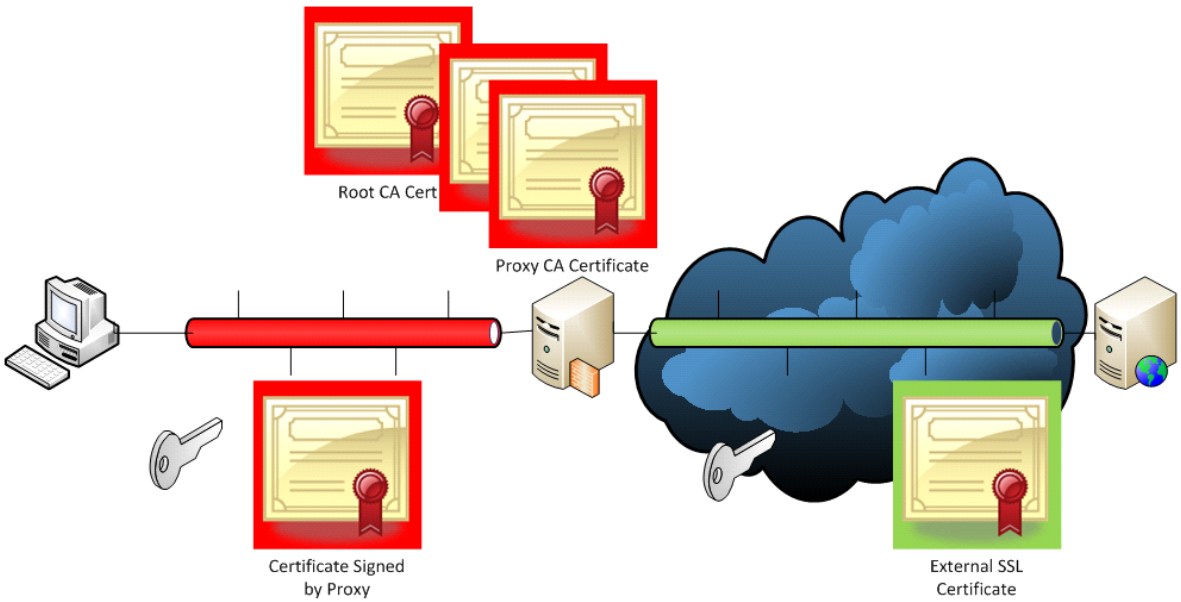


Fig. 2 – SSL Proxy Configuration.

Effectively they implement a man in the middle (MITM) attack. SSL Proxy certificates are functionally the same as intermediate CA certificates. They allow the SSL proxy to intercept the server certificate, replace the key with one known by the proxy, and then resign the certificate. This resigned certificate is presented to, and trusted by the browser. The proxy generates a second SSL connection outbound to the original web server.

Their use has been relatively common for several years, but they have been slowly gathering negative publicity since early 2012 when Trustwave admitted issuing an SSL proxy certificate [9]. Mozilla published a series of letters to participants of their trusted root certificate program offering a period of amnesty instead of revoking the Trustwave Root CA Certificates [10] and an opportunity for them to revoke the “SSL Spying Certificates” [11].

2.6b. TurkTrust

More recently, in what can be better described as a series of unfortunate events instead of malicious activity, the certificate services provider TurkTrust was discovered to have issued intermediate CA certificates to two subscribers. Normal SSL certificates had been requested. One of the subscribers immediately requested the delivered certificate be revoked. The second certificate was used to establish an SSL proxy in late 2012 [12]. Due to this incident the CA certificates have been removed from the browser and TurkTrust has been suspended from the trusted CA certificate program by Mozilla [13].

The chrome browser implements security mechanisms beyond those specified in the X.509 series of standards. This browser implements white listing of SSL public keys for the google.com domain. As TurkTrust was not one of the providers of valid google.com certificates, the chrome browser flagged the TurkTrust certificate as invalid [14].

Ultimately the public exposure of a proxy google.com certificate would put any CA which issued the proxy certificate at risk of blacklisting in several browsers.

3. CRYPTO SECURITY

3.1. Public Key Enabled Malware

The following is not intended to be a detailed analysis of the listed malware. The intent is to identify the manner in which the malware obtained a valid digital signature. The lessons from these malware samples are to limit your trust where practical on critical infrastructure, and to not become the source for signatures on the next generations of malware.

3.1a. Stuxnet

The Stuxnet malware became public in July 2010. Lost in the media coverage speculating on the purpose and authors of this malware was the digital signature which is one of the stealth enabling features. The malware includes two files which are installed as signed drivers by the operating system. The signature allows them to be installed without generating a prompt to authorize the installation. The initial drivers are signed with certificates and keys stolen from Realtek and JMicron [15]. More information regarding the pervasiveness of digitally signed malware is provided at [16]. Finally the Symantec analysis of Stuxnet is provided at [17] which interestingly state the risk level as low.

3.1b. Duqu

The Duqu virus, while carrying a substantially different payload to the Stuxnet virus shares several similarities including the digitally signed driver with a certificate from C-Media. The Laboratory of Cryptography and system Security (CrySyS) at the Budapest University of Technology and Economics has published a detailed analysis of Duqu [18]. A further article quoting the McAfee assessment is published at [19] and again an analysis by Symantec at [20].

3.1c. Flame

The Flame malware took an alternative approach to signing malware. Instead of stealing Certificates and keys, Flame is reported to have implemented a MD5 chosen prefix collision attack against a Microsoft certificate which supported code signing [21]. In response to this event Microsoft has moved the two relevant certificates to the untrusted certificates container [22].

3.1d. 5000 More signed by Adobe

Again in 2012, a compromised server holding a code signing certificate was substantially abused to sign more than 5000 pieces of malware. When initially disclosed, [23]. Adobe downplayed the attack as only enabling the signing of two pieces of malware. Further analysis by other security experts confirmed the scope and highlighted the technical knowledge required to implement the attack [24].

3.2. Attacks against Algorithms

3.2a. Timing Attacks

Timing attacks were already well documented in the mid 1990's at a time when PKI was still in its infancy. General attacks were documented on several discrete logarithmic algorithms including RSA, DSA, and Diffie-Hellman. By 2003 research had matured to the point where controlled implementations were being tested on academic networks against Open-SSL implementations [25]. One implementation of this attack works by measuring the time taken to complete known mathematical operations. The attacker measures the timing differences caused when different information is submitted to the SSL Server. The exact mechanism here is extra reductions during Montgomery reduction for a manipulated input by the attacker. At the time, the technique was being successfully implemented (albeit in a very controlled environment) for key lengths up to 1024 bits.

Blinding, an acceptably efficient mitigation for this attack was already envisioned in much earlier papers [26], but for implementations such as Open-SSL, this functionality was still not enabled by default in 2003. Blinding is a process where an additional calculation with random data is added to obfuscate the timing differences.

3.2b. Short key lengths

In the fall of 2011 compromised certificates based on RSA-512 were detected from a Malaysian Certificate Authority [27]. The commonly agreed attack vector was the factoring of the keys. Microsoft made a tool available to remove trust in RSA-512 certificates in 2012 [28]. The solution here is obvious: follow the attack trends to ensure that you are able to upgrade key lengths before you are forced into an emergency re-key by an attacker factoring keys.

3.2c. Hash collisions

MD5 based hash collisions have been shown to be an implemented reality by malware such as Flame, but the current discussions still speculate on when the theoretical collisions against SHA-1 will arrive. Schneier [29] calculates that collisions will become affordable by 2018. As we have seen, a single chosen plain text hash collision can result in a substantial amount of damage. Solution here is the same as for short key lengths.

3.3. Attacks against Cryptographic Implementations

3.3a. Random Number Generator (RNG)

Over the years there have been many flawed implementations of random number generators, including implementations by Microsoft and Netscape to name just a few. More recently, researchers have uncovered a significant number of active RSA keys which are either default keys duplicate keys, or contain a shared prime factor [30]. Some researchers immediately came to the conclusion that RSA was broken [31], but there are several other causes for weak keys.

Default keys exist due to poor configuration practices where keys included in the device from the factory are not replaced prior to in-servicing the equipment.

Keys that are duplicates or that share a prime factor result from random number generators with low entropy [32]. Duplicate key pairs likely are the result of both primes being generated with poor entropy. The last case (shared prime factor) likely results from the first prime number resulting from poor entropy in the RNG, but the second prime is generated after the RNG is re-seeded to improve entropy. For completeness, the bugs discovered in the Microsoft and Netscape RNG implementations are [33] and [34]. Formal security evaluations are the best protection against flawed RNG implementations.

4. CONCLUSIONS

This paper has only scratched the surface of the majority of these threats, but it provides plenty of links for further analysis. The threats and the suggested countermeasures provide a broad threat landscape to use

as an input to a risk assessment process. The risk assessment process is critical in determining the security services and security mechanisms required to secure PKI implementations.

The proliferation of signed malware is here to stay. This paper provides one final lesson; learn the lessons of others before you learn them from scratch by yourself.

5. DISCLAIMER

Any opinions expressed herein do not necessarily reflect the views of the NCI Agency, NATO and the NATO nations, but remains solely those of the author.

REFERENCES

1. Fox-IT BV, *Black Tulip Report of the investigation into the DigiNotar Certificate Authority breach*, 13 August 2012, Online, Available: http://www.rijksoverheid.nl/documenten-en-publicaties/rapporten/2012/08/13/black-tulip-update.html.
2. G. L. Guerin, *Microsoft, Verisign, and Certificate Revocation*, 13 May 2001, Online. Available: http://www.amug.org/~glguerin/opinion/revocation.html.
3. Microsoft, *MS01-017: Erroneous VeriSign-Issued Digital Certificates Pose Spoofing Hazard*, 21 February 2007, Online. Available: http://support.microsoft.com/kb/293818.
4. B. Delpy, *mimikatz*, Online. Available:4. http://blog.gentilkiwi.com/mimikatz. [Accessed 2013].
5. Microsoft, *Microsoft Security Advisory (2524375)*, 6 July 2011, Online, Available: http://technet.microsoft.com/en-us/security/advisory/2524375.
6. G. Keizer, *Solo Iranian hacker takes credit for Comodo certificate attack*, 27 March 2011, Online, Available: http://www.computerworld.com/s/article/9215245/Solo_Iranian_hacker_takes_credit_for_Comodo_certificate_attack.
7. RSA FraudAction Research Labs, *Anatomy of an Attack*, 1 April 2011, Online, Available: http://blogs.rsa.com/anatomy-of-an-attack/.
8. K. Zetter, *RSA Agrees to Replace Security Tokens After Admitting Compromise*, Wired, 6 July 2011, Online, Available: http://www.wired.com/threatlevel/2011/06/rsa-replaces-securid-tokens/.
9. L. Constantin, *Trustwave Admits Issuing Man-in-the-Middle digital Certificate, Mozilla Debates Punishment*, CIO, 8 Febrary 2012, Online, Available: http://www.cio.com/article/699762/Trustwave_Admits_Issuing_Man_in_the_Middle_Digital_ Certificate_Mozilla_Debates_Punishment.
10. L. Constantin, *Mozilla Will Ask All Certificate Authorities to Revoke SSL-Spying Certificates*, CIO, 14 February 2012, [Online]. Available: http://www.cio.com/article/700490/Mozilla_Gives_CAs_a_Chance_to_Come_Clean_About_Certificate_Policy_Violations.
11. L. Constantin, *Mozilla Gives CAs a Chance to Come Clean About Certificate Policy Violations*, CIO, 20 February 2012, Online, Available: http://www.cio.com/article/700161/Mozilla_Will_CAs_a_Chance_to_Come_Clean_About_Certificate_Policy_Violations.
12. P. Ducklin, *The TURKTRUST SSL certificate fiasco - what really happened, and what happens next?*, Sophos, 8 January 3012, Online, Available: http://nakedsecurity.sophos.com/2013/01/08/the-turktrust-ssl-certificate-fiasco-what-happened-and-what-happens-next/.
13. M. Coates, *Revoking Trust in Two TurkTrust Certificates*, Mozilla, 3 January 2013, Online, Available: http://blog.mozilla.org/security/2013/01/03/revoking-trust-in-two-turktrust-certficates/.
14. A. Langley, *Imperial Violet: Public key pinning*, 4 May 2011, Online, Available: http://www.imperialviolet.org/2011/05/04/pinning.html.
15. M. Russinovich, *Analyzing a Stuxnet Infection with the Sysinternals Tools, Part 1*, Microsoft, 30 March 2011, Online, Available: http://blogs.technet.com/b/markrussinovich/archive/2011/03/30/3416253.aspx.
16. L. Constantin, *Digitallly Signed Malware Is Increasingly Prevalent, Researchers Say*, IDG News Service, 15 March 2012, Online, Available: http://www.pcworld.com/article/251925/digitally_signed_malware_is_increasingly_prevalent_ researchers_say.html.
17. Symantec, *W32.Stuxnet*, 26 February 2013, Online, Available: http://www.symantec.com/security_response/writeup.jsp?docid=2010-071400-3123-99.
18. Laboratory of Cryptography and System Security (CrySyS), "*Duqu: A Stuxnet-like malwere found in the wild*," 11 October 2011, Online, Available: http://www.crysys.hu/publications/files/bencsathPBF11duqu.pdf.
19. T. Espiner, *McAfee: Why Duqu is a big deal*, ZDNet, 26 October 2011, Online, Available: http://www.zdnet.com/mcafee-why-duqu-is-a-big-deal-3040094263/.
20. Symantec, *W32.Duqu: The precursos to the next Stuxnet*, 23 November 2011, Online, Available: http://www.symantec.com/ content/en/us/enterprise/media/security_response/whitepapers/w32_duqu_the_precursor_to_the_next_stuxnet.pdf.
21. M. Stevens, *Technical Background of the Flame Collision Attack*, Centrum Wiskunde & Informatica, 7 June 2012, Online, Available: http://www.cwi.nl/news/2012/cwi-cryptanalist-discovers-new-cryptographic-attack-variant-in-flame-spy-malware.
22. Microsoft, *Microsoft Security Advisory (2718704) Unauthorized digital Certificates could Allow Spoofing*, 3 June 2012, Online, Available: http://technet.microsoft.com/en-us/security/advisory/2718704.

23. D. Goodin, *Adobe to revoke crypto key abused to sign malware apps (corrected)*, Ars Technica, 28 September 2012, Online, Available: http://arstechnica.com/security/2012/09/adobe-to-revoke-crypto-key-abused-to-sign-5000-malware-apps/.
24. R. NarinE, *Adobe code signing infrastructure hacked by 'sophisticated threat actors'*, ZDNet, 27 September 2012, Online, Available: http://www.zdnet.com/adobe-code-signing-infrastructure-hacked-by-sophisticated-threat-actors- 7000004925/ ?s_cid=e539.
25. D. Brumley and D. Boneh, *Remote Timing Attacks are Practical*, in USENIX, Washington, 2003.
26. P. C. Kocher, *Timing Attacks on Implementations of Diffie-Hellman, RSA, DSS, and Other Systems*, in Advances in Cryptology - CRYPTO '96, 16th Annual International Cryptology Conference, Proceedings, Santa Barbara, 1996.
27. M. Sandee, *RSA 512 Certificates abused in the wild*, 21 11 2011, Online, Available: https://www.fox-it.com/en/blog/rsa-512-certificates-abused-in-the-wild/.
28. Microsoft, *Microsoft Security Advisory (2661254) Update For Minimum Certificate Key Length*, 14 August 2012, Online, Available: http://technet.microsoft.com/en-us/security/advisory/2661254.
29. B. Schneier, *When Will We See Collisions for SHA-1*, 5 October 2012, Online, Available: http://www.schneier.com/blog/archives/2012/10/when_will_we_se.html.
30. D. Auerbach and P. Eckersley, *Researchers Use EFF's SSL Observatory To Discover Widespread Cryptographic Vulnerabilities*, Electronic Frontier Foundation, 14 February 2012, Online, Available: https://www.eff.org/rng-bug.
31. A. K. Lenstra, J. P. Hughes, M. Augier, J. W. Bos, T. Kleinjung and C. Wachter, *Ron was wrong, Whit is right*, 14 February 2012, Online, Available: http://eprint.iacr.org/2012/064.pdf.
32. N. Heninger, *New research: There's no need to panic over factorable keys - Just mind your Ps and Qs*, Online, Available: https://freedom-to-tinker.com/blog/nadiah/new-research-theres-no-need-panic-over-factorable-keys-just-mind-your-ps-and-qs/.
33. L. Dorrendorf, Z. Gutterman and B. Pinkas, *Cryptanalysis of the Random Number Generator of the Windows Operating System*, 4 November 2007, Online, Available: http://eprint.iacr.org/2007/419.pdf.
34. I. Goldberg and D. Wagner, *Randomness and the Netscape Browser*, January 1996, Online, Available: http://www.cs.berkeley.edu/~daw/papers/ddj-netscape.html.

Received May 20, 2013

THE PUBLISHING HOUSE
OF THE ROMANIAN ACADEMY

PROCEEDINGS OF THE ROMANIAN ACADEMY, Series A,
Volume 14, Special Issue 2013, pp. 295–302

GENERATING CHAOTIC SECURE SEQUENCES USING TENT MAP AND A RUNNING-KEY APPROACH

Adriana VLAD[1,2], Adrian LUCA[1], Octavian HODEA[1], Relu TATARU[1]

[1]Faculty of Electronics, Telecommunications and Information Technology,
POLITEHNICA University of Bucharest, 1-3, Iuliu Maniu Bvd. Bucharest 6, Romania
[2]The Research Institute for Artificial Intelligence,
Romanian Academy, 13, Calea 13 Septembrie, Bucharest 5, Romania
Corresponding author: Adriana VLAD, E-mail: avlad@racai.ro

This paper completes recent results obtained by extending the running-key cipher procedure from natural language to applications over chaotic systems. We apply the new running-key approach on the chaotic tent map and prove its utility in obtaining practically zero-redundant pseudo-random number generators, alongside with the possibility to consider the initial condition and the tent map control parameter as elements in the secret key – a desideratum in chaos-based cryptography. The results are both theoretically and experimentally supported by combining concepts from information theory and statistical methods in the context of the chaotic system. The statistical evaluation includes NIST test suite for testing the randomness of the proposed binary generator. Based upon the results presented in this paper, the provided generator can be used for designing new cryptosystems where the pseudo-random binary sequences can be a chief support.

Key words: tent map, pseudorandom binary sequences, running-key cipher, noisy channel, NIST test suite

1. INTRODUCTION

The running-key procedure advanced in [1] and [2] for the logistic function is here extended and adapted for tent map. Mathematically speaking, tent map is a discrete time chaotic system described by relation (1):

$$x_{n+1} = f(x_n) = \begin{cases} \dfrac{x_n}{a} & ,0 \le x_n \le a \\ \dfrac{1-x_n}{1-a} & ,a < x_n \le 1 \end{cases} \tag{1}$$

where, $a \in [0,1]\backslash\{0.5\}$ is the control parameter and x_n is the current state of the system. Tent map defined in (1) has uniform invariant probability density in [0, 1] interval. Binary sequences (further denoted by Z) are obtained from the real x_n values of tent map by a comparison with a c threshold equal with the control parameter a as in Fig. 1 and relation (2).

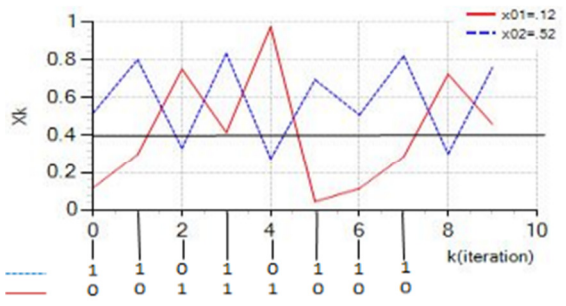


Fig. 1 – Two binary sequences generated by tent map for two different initial conditions. Illustration for $c = a = 0.4$.

$$z_k = \begin{cases} 0, & 0 \le x_k \le c \\ 1, & c < x_k \le 1 \end{cases} \quad (2)$$

The working versions of running-key procedure are illustrated in Fig. 2.

$$\begin{aligned} Z_0 + Z_1 &\Longrightarrow Y_1 \\ Z_0 + Z_1 + Z_2 &\Longrightarrow Y_2 \\ Z_0 + Z_1 + Z_2 + Z_3 &\Longrightarrow Y_3 \\ \vdots & \\ Z_0 + Z_1 + Z_2 + Z_3 + \dots + Z_k &\Longrightarrow Y_k \end{aligned}$$

Fig. 2 – A running-key approach applied to the chaotic system. Z_i stands for the binary typical sequences.

The method has the particularity that all the summed sequences are typical binary sequences generated by tent map (1) considering successive iterations for a fixed a control parameter and a binarization threshold $c = a$. The summations in the Fig. 2 are bit by bit modulo 2.

The extending of running-key procedure for chaotic systems is due to the ergodicity property of these systems. For fixed a parameter in (1) we have an ergodic random process. Each parameter change will lead to another random process. Thus, future discussions will be made for fixed a value of the parameter. Each typical sequence will be defined by a randomly chosen initial condition in $[0,1]$ and a fixed control parameter $c = a$, the same for all the summed sequences.

Our particular concern when applying the running-key approach (see Fig. 2) is to evaluate the number of summed typical binary sequences that can lead to Y_k sequence compatible with the fair coin model. In other words, we aim to determine the k value, so that the output information source that produces Y_k sequences becomes a binary memory-less source of zero-redundancy. In this case, the Y_k sequence will be statistically independent of its components (sequences that are summed up) and it would not be possible to be decomposed into its components.

This way of proceeding, that we call the running-key approach, includes a representation of variants in Fig. 2 by using a cascade of information channels that allows a simple way to determine the value of k when reaching the Y_k sequence compatible with the fair coin model. Using the cascade of information channels, all evaluations will be done directly inspecting the output sequence Y_k. The requirement that Y_k is compatible with the fair coin model implicitly ensures the fact that Y_k cannot be separated into its components. This condition is a desideratum that can be reached (verified) by various statistical tests carried on Y_k, but avoiding more complicated assessments of "cryptanalysis" type.

Note: The term "cryptanalysis" is misused here because what we get in the end is a good quality statistical pseudo-random binary generator, where Y_k sequences could be used as enciphering sequences (keys) in a cryptosystem. For a better understanding, recall that the evaluation of running-key cipher applied to natural language was based on cryptanalysis. Thus, in versions 1 and 2 (corresponding to Fig. 2) the cipher was broken (*i.e.* the cryptograms Y_1 and Y_2 could be decomposed into the natural texts that participated to the respective summation), but in the variant 3 (*i.e.* Y_3, meaning four summed natural texts) the cipher could not be broken, result which allowed assessment of relative redundancy (about 75%) for English language, [1] - [6].

In what follows, we show that the running-key procedure applied on tent map will provide a binary pseudo-random number generator compatible with the fair coin model for a large range of values for a control parameter.

Section 2 presents the theoretical evaluation of the generator obtained by the running-key approach. Section 3 shows the experimental results obtained when catching the k value for which Y_k type sequence corresponds to fair coin model. The investigation was made using probability tests supported by a Monte-Carlo analysis and by the NIST test suite. Section 4 presents final remarks and conclusions.

2. THEORETICAL EVALUATION

By successively iterating the tent map (1) and choosing binarization threshold equal to tent map parameter, the obtained binary sequence obeys to the *i.i.d.* statistical model (data coming up from independently and identically distributed random variables), result shown in [7]. The result is valid for any control parameter value (except $a = 0.5$, when the tent map statistical behavior is no more chaotic). Although binary data are statistically independent even in successive iterations, in order to comply with the fair coin model a numerical restriction concerning the parameter range of values is required [7] and [8]. Thus, if the parameter is chosen in the interval (0.49995, 0.50005), a binary pseudorandom generator of good statistical quality can be obtained for typical sequences of length at most equal to 10^6 binary symbols (see section 3). Although being of a good statistic quality, the generator has the disadvantage that the initial condition can be easily recovered based on a binary sequence. In this case, the initial condition and the control parameter from (1) may not be included in the secret key in cryptographic applications, [8] and [9].

By the running-key procedure, Fig. 2, we get Y_k type binary sequences compatible with the fair coin model for a wide range of values of the control parameter. According to the running-key procedure, the summed sequences will not be recovered from Y_k. In this way a major obstacle against the recovery of the control parameter value and of the initial conditions will be encountered, which could lead to their inclusion in the secret key.

In Fig. 2, Z_i are *i.i.d.* binary sequences of *N* size, obtained by considering all successive iterations of (1), fixed a value and $c = a$.

Fig. 3 introduces a new view of the running-key procedure from Fig. 2, through a cascade of information channels. The cascaded information channels allow to catch the moment when the running-key procedure stops, by evaluating the entropies of the secondary sources $Y_1, Y_2, \cdots Y_k$. All the implied information sources $Z_0, Y_1, Y_2, \cdots Y_k$, are binary and memory-less; Fig. 3 allows a decision concerning the degree of dependence/independence between the Z_0 input source (which corresponds to the tent map and to its typical sequences Z_i) and the Y_k output source.

Note: The typical sequence $Y_1 = Z_0 + Z_1$ (from Fig. 2) can be viewed as a particular realization of the Y_1 source in Fig. 3 (output of the first information channel). Similarly, Y_k typical sequence from Fig. 2 can be viewed as a particular realization of the Y_k source in Fig. 3 (Y_k source in Fig. 3 and Y_k typical sequence from Fig. 2 are denoted by the same letter).

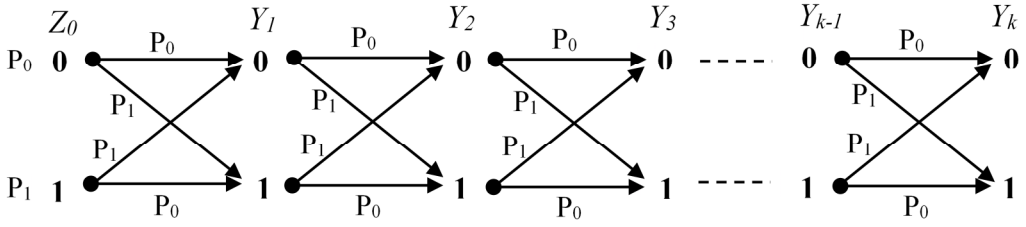


Fig. 3 – An information channel depiction of running-key approach.

The cascaded information channels are binary symmetric channels, each of them having the noise matrix from relation (3). P_{ij} elements of the noise matrix are actually the probabilities of the Z_0 information source, $P_0 = P(Z_0 = 0) = a$ and $P_1 = P(Z_0 = 1) = 1 - a$ [7]. The Y_1 secondary source has the probabilities: $P(Y_1 = 0) = a^2 + (1-a)^2 = 1 - 2a(1-a)$ and $P(Y_1 = 1) = 2a(1-a)$. It follows that the entropy assigned to Y_1 binary source can be obtained by using the entropy function $H(p) = -p\log_2 p - (1-p)\log_2(1-p)$. So $H(Y_1) = H(2a(1-a))$.

	P_{ij}	Y_1: 0	Y_1: 1
Z_0	0	a	$1-a$
	1	$1-a$	a

$$P_{ij} = P(Y_1 = j / Z_0 = i), \quad i, j \in \{0,1\} \tag{3}$$

Table 1 presents the entropy values corresponding to Z_0 and Y_k information sources, evaluated for two control parameter values: $a = 0.2$ and $a = 0.4$.

Table 1

Entropies assigned to the cascaded information channels

Entropy	$a = 0.2$	$a = 0.4$
$H(Z_0) = H(a)$	0.7243	0.9699
$H(Y_1) = H(2a(1-a))$	0.9074	0.9987
$H(Y_3) = H(2u(1-u));\ u = 2a(1-a)$	0.9915	0.9994
$H(Y_5) = H(2w(1-w));\ w = 2u(1-u)$	0.99978	0.9999999

Based on Fig. 3 and fundamentals from information theory and cryptography [4], [5], [6] and [10], the cascaded binary symmetric channels successively convey information from the Z_0 input source, which is a redundant source, to the Y_k which is practically a non-redundant source. Thus we have the following relationship between the entropies of the binary successive sources:

$$H(Z_0) < H(Y_1) < H(Y_2) < H(Y_3) < \cdots < H(Y_k) \tag{4}$$

The running-key procedure determines the k value for which $H(Y_k) \cong 1$. To decide when $H(Y_k) \cong 1$, a probability test on the Y_k binary sequence will be used. The decision that the sequence is compatible with the fair coin model will be taken based on type II statistical error [11].

The probability test has the following hypotheses:

- ✓ null hypothesis, H_0: $p = p_0$
- ✓ alternative hypothesis, H_1: $p \neq p_0$

where p is the true probability of the investigated event.

The statistical significance level is $\alpha = 0.05$, thus $z_{\alpha/2} = 1.96$ ($z_{\alpha/2}$ is $\alpha/2$-point value of the standard Gaussian law). The H_0 hypothesis is accepted if $|\hat{p} - p_0| \leq \varepsilon = z_{\alpha/2}\sqrt{p_0(1-p_0)/N}$ where $\hat{p}$ is the estimated value of p probability and N is the size of the $i.i.d.$ sequence. The type II statistical error probability for the test is expressed by (5).

$$\beta(N,\delta) = \int_{p_0-\varepsilon}^{p_0+\varepsilon} \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{(x-\tilde{p})^2}{2\sigma^2}\right) \mathrm{d}x \tag{5}$$

$$\varepsilon = z_{\alpha/2}\sqrt{\frac{p_0(1-p_0)}{N}},\ \ \tilde{p} = p_0(1 \pm \delta),\ \sigma^2 = \tilde{p}(1-\tilde{p})/N$$

The decision on k value ($H(Y_k) \cong 1$) relies on type II statistical error probability and depends upon two parameters, N and δ (here $p_0 = 0.5$). In order to be sure that the binary sequence always obeys fair coin model (i.e. the final expected result), δ needs to be reevaluated for any N length (required by the application).

The β type II statistical error probability of accepting wrong data as good data (*i.e.* to accept H_0 even if the coin is slightly unbalanced) is evaluated according to relation (5) where $\alpha = 0.05$ and $p_0 = 0.5$. For example if $\delta = 0.001$ and $N \leq 65536$, the type II statistical error probability is greater than 0.94. For a data volume $N \leq 10^6$ (needed by NIST tests) and aiming that the probability test will pass in about 95% of cases ($\beta \approx 0.95$), it results $\delta \approx 10^{-4}$. If follows that a parameter value chosen in the interval [0.49995, 0.50005] will enable to obtain a typical Y_k binary sequence complying with the fair coin model without any summation.

By applying the running-key procedure for a choice of the a parameter value in a larger interval, *e.g.* $a \in [0.4, 0.6]$, the decision that the output sequence complies with the fair coin model is taken when $k = 4$ respectively for Y_4 (summing 5 typical sequences generated by tent map), see *Table 1* and Section 3.

Comments on Table 1:

For $a = 0.2$ parameter value, the resulting entropy of the input binary source is $H(Z_0) = 0.7243$ bits. If we sum two typical Z_i sequences emitted by tent map it results Y_1 information source of entropy $H(Y_1) = 0.9074$ bits. If we sum six typical sequences we obtain Y_5 sequence emitted by a practically non-redundant information source namely, $H(Y_5) = 0.99978$.

Following the reasoning applied for the evaluation of natural language redundancy, [2], [3] and [6], the recovery of the 6 sequences of N length that participated to the summation would imply to determine $6 \cdot N \cdot H(Z_0)$ unknown binary symbols from N known binary symbols (representing Y_k).

Obviously, we cannot decompose Y_5 into the 6 components. If we could decompose it into the 6 components, we should have $6 \cdot H(Z_0) < 1$. Judging from this, it would be enough to make a summation of 2 typical sequences produced by tent map with $c = a = 0.2$. In order to obtain a non-redundant generator for $a = 0.2$ and $N = 10^6$, we need more than 5 summations for the needed accuracy. The 5 summations implicitly ensure statistical independence between the Y_5 sequence and any of the Z_i sequences that participated to the summation, result which is valid only for N values (*e.g.* $N < 65536$) smaller than the requirement of the NIST test suite.

To conclude: Finally we obtain a non-redundant sequence, completely independent of any typical sequence participating in the summation. So, it is difficult to believe that someone could easily recover, without exhaustive trials, the $k + 1$ initial conditions and the control parameter value, based on Y_k.

3. EXPERIMENTAL RESULTS

Here we present experimental results on the proposed generator. The running-key procedure is evaluated by successively investigating the typical sequences $Y_1, Y_2, \cdots, Y_k$, aiming to catch the moment when Y_k complies with the fair coin model. Section 3.1 presents the experimental study by using basic statistical tests concerning *m*-gram probability (m successive binary symbols) supported by the Monte-Carlo analysis. Section 3.2 presents experimental studies on the same generator using the NIST test suite.

3.1. Basic statistical methods

For this generator, the control parameter is equal to the binarization threshold, in which case the Z_i sequences obey to the *i.i.d.* model, however their probability law is not uniform. By successive summations we aim to determine the *k* value when Y_k sequence complies with the fair coin model.

Y_k sequences are obtained by a bit by bit modulo 2 summation of the Z_i sequences. Sequences denoted by Z_i are of size $N = 10^6$ bits and are generated by (1) for a fixed a control parameter value and randomly chosen initial conditions according to the uniform law in the (0, 1) interval. For example, summing up the two data sequences $Z_0 \leftrightarrow (z_{01}, z_{02}, \cdots z_{0N})$ and $Z_1 \leftrightarrow (z_{11}, z_{12}, \cdots z_{1N})$ we obtain $Y_1 \leftrightarrow (y_{11}, y_{12}, \cdots y_{1N})$, where $y_{1j} = z_{0j} + z_{1j}$.

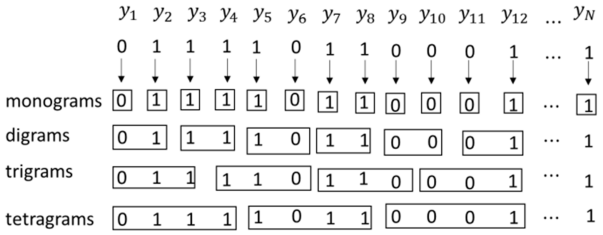


Fig. 4 – The *m*-gram experimental data sets.

Based on the obtained Y_k binary sequences we evaluated the *m*-gram ($m = 1,2,3,4$) probability by applying the usual probability test with the following hypotheses: H_0, meaning that the investigated probability is equal to the p_0 expected theoretical value; H_1, the investigated probability differs from p_0. The H_0 hypothesis is accepted if $|\hat{p} - p_0| \leq \varepsilon = z_{\alpha/2}\sqrt{p_0(1-p_0)/N}$ where $\hat{p}$ is the estimated value of the investigated *m*-gram probability. The *m*-gram data sets submitted to test are extracted from Y_k as shown in Fig. 4. Note that the data set volume for monogram is $N = 10^6$, for digram is $N/2$, for trigram $N/3$ and for tetragram $N/4$. The corresponding p_0 theoretical probabilities are: 0.5 for monogram, 0.25 for digram, 0.125 for trigram and 0.0625 for tetragram. We made a detailed analysis for each possible *m*-gram ($m = 1,2,3,4$).

For each and every investigated m-gram, we applied a Monte-Carlo analysis by resuming the probability test 500 times and recording the proportion of the H_0 hypothesis acceptance. The decision that Y_k sequence complies with the fair coin model was taken if the recorded proportion was within the interval $[0.93, 0.97]$, for each and every investigated m-gram.

In *Table 2* we show this kind of results for $a = 0.4$ and for the k summation number equal to 3 and 4. It can be noticed that the proportions of the H_0 hypothesis acceptance is inside $[0.93, 0.97]$ for all m-grams, when $k = 4$.

Table 2

Proportion of H_0 acceptance for m-gram probability test for $a = 0.4$

m-gram	Y_3	Y_4	m-gram	Y_3	Y_4
0	0.622	0.96	011	0.936	0.958
1	0.622	0.96	100	0.944	0.948
00	0.742	0.974	101	0.93	0.934
01	0.95	0.95	110	0.95	0.942
10	0.942	0.94	111	0.912	0.96
11	0.716	0.96	0000	0.884	0.96
000	0.834	0.948	0001	0.926	0.962
001	0.946	0.936	0010	0.926	0.942
010	0.942	0.956	1111	0.864	0.966

We made similar investigations as in *Table 2* for various a control parameter values. The overall study was summarized in *Table 3*. Namely, for each k value we determined a range of values for the choice of a control parameter. If the a control parameter is chosen inside the respective interval, the corresponding Y_k sequence complies with the fair coin model. For example, for $k = 3$ the a parameter value should be chosen within $(0.43, 0.57)$ interval.

Table 3

Tent map control parameter intervals that enable to generate the Y_k non-redundant sequences

k	Y_k	Assigned interval for a
0	$Y_0 = Z_0$	(0.49995, 0.50005)
1	Y_1	(0.497, 0.503)
2	Y_2	(0.48, 0.52)
3	Y_3	(0.43, 0.57)
4	Y_4	(0.39, 0.61)

3.2. NIST test suite

The performances of the proposed chaotic generator were evaluated using one of the most popular standards for investigating the randomness of binary data, namely NIST (National Institute of Standards and Technology) test suite. The NIST statistical test suite was specially designed for randomness examination of hardware or software data generated from cryptographic random or pseudo-random generators.

NIST tests were primarily applied to verify the results obtained by the running key approach, testing the Y_k sequences from sub-section 3.1. Thus, we checked the ranges of the control parameter a for which the chaotic generator provides pseudo-random and non-redundant binary sequences Y_k.

For our investigation we have generated a number of $m = 1000$ different Y_k binary sequences for each k value. Each Y_k sequence was generated using $k + 1$ initial conditions for tent map, uniformly chosen in the range $(0,1)$, each of length 10^6 bits (fixed a parameter and $c = a$).

For each Y_k sequence, all 15 variants of NIST statistical tests (divided in 188 tests) were applied. Each test provides a p-value [12]. Similarly to other statistical tests, the NIST tests are based on hypotheses testing. Hypotheses testing is a procedure to determine whether an assumption about a particular theory is reasonable. In this case, the assumption is that a certain sequence of 0 and 1 is random [12].

To validate each statistical test and to determine its passing ratio, a significance level (α) is firstly chosen. A value $\alpha = 0.01$ of the significance level assumes that 99% of the sequences should pass the

statistical test. In our experiments the significance level was fixed to 1%. By the probability estimation theory and resuming each test 1000 times, the range of acceptable proportions for each type of test can be determined by using the confidence interval defined as $(p \mp 3\sqrt{p(1-p)/m}$, where $p = 1 - \alpha = 0.99$ and $m = 1000$. Thus, the passing ratio for the Y_k sequences is restricted to the acceptance interval $[0.9806, 0.9994]$.

Note: In this paper only 15 of the 188 NIST statistical tests are illustrated. For tests existing in several variants only the results for one randomly chosen test were provided for illustration. These categories include: Cumulative Sums, Non Overlapping Template, Random Excursions, Random Excursions Variant and Serial statistical tests.

The randomness of Y_k sequences generated using the running-key approach was investigated using the NIST statistical test suite in order to verify/validate the range of a control parameter according to *Table 3*. For these intervals we claim that Y_k sequences follow closely the fair coin model. The margins (left/right limit) of each interval were tested along with several values inside the range. The results confirmed the high quality for the provided intervals. *Table 4* illustrates one of the NIST analysis for $k = 4$ (corresponding to Y_4) and the control parameter $a = 0.4210$ and threshold $c = a$.

Table 4

NIST results for the proposed generator ($c = a = 0.4210$)

C1	C2	C3	C4	C5	C6	C7	C8	C9	C10	P-val.	P.R.%	Statistical Test
111	107	107	100	96	95	101	88	102	93	0.86	0.989	**Frequency**
82	92	111	110	100	90	110	101	103	101	0.51	0.989	**BlockFrequency**
115	100	103	84	117	83	103	87	110	98	0.14	0.986	**CumulativeSums**
99	80	106	85	98	122	83	104	108	115	0.04	0.992	**Runs**
94	90	104	102	97	92	98	108	112	103	0.88	0.987	**LongestRun**
110	114	88	91	96	93	106	93	100	109	0.59	0.986	**Rank**
106	90	101	113	87	107	110	88	94	104	0.51	0.989	**FFT**
100	91	109	98	116	91	103	109	92	91	0.60	0.987	**NonOverlappTemplate**
117	103	101	88	90	108	100	86	105	102	0.50	0.985	**OverlappTemplate**
122	97	108	84	100	95	88	110	103	93	0.25	0.986	**Universal**
96	100	88	102	109	107	93	117	87	101	0.53	0.987	**ApproximateEntropy**
58	54	71	64	67	51	64	65	52	59	0.65	0.998*	**RandomExcursion**
58	61	61	57	50	60	69	49	73	67	0.45	0.995*	**RandomExcursionV**
105	91	103	88	117	103	87	103	107	96	0.53	0.989	**Serial**
124	96	96	85	94	100	93	88	111	113	0.14	0.987	**LinearComplexity**

* RandomExcursions and RandomExcursionsVariant tests used fewer sequences to complete the tests

Interpretation of Table 4

The proposed chaotic generator was analyzed using the latest version of the NIST statistical test suite published on the official website of the National Institute of Standards and Technology. The result provided by NIST consists in a test report for each binary file of length $N \times m$ bits under investigation. The resulting test report contains a table that includes the following columns:

- ✓ STATISTICAL TEST column presents the names of the tests
- ✓ PROPORTION (Passing Ratio – P.R.) column represents the passing ratio of the Y_k sequences from a total of m sequences. This value should be in the range provided in this section in case of success
- ✓ $C_1, C_2, C_3, \cdots, C_{10}$ columns represent the number of sequences (from m) for which the p-value is located in one of the 10 disjoint intervals of length 0.1 which cover the [0,1] range
- ✓ P-value column indicates the uniformity of the p-values obtained for each statistical test [11]. This can be considered as a P-value of p-values and along with the proportion column it represents an indicator of the statistical test success. This P-value is obtained as follows:

  – The chi-square (χ^2) test is applied to measure the spreading of the p-values over [0,1] using the results provided by columns C_i, $(i = 1,2, \dots 10)$

  – The value $P\_value_T = igamc(\frac{9}{2}, \frac{\chi^2}{2})$ is evaluated. If $P\_value_T \geq 0.0001$, the p-values can be considered uniformly distributed in [0,1].

According to NIST statistical test suite, regarding the passing ratio and the uniformity of p-values obtained after applying the statistical tests, the proposed chaotic generator is eligible to be used in pseudo-random binary data generation, with direct applications in cryptography.

4. CONCLUSIONS

In this study we applied the new running-key method on the chaotic tent map and demonstrated its utility in designing non-redundant pseudo-random number generators, alongside with the possibility to consider the initial condition and the tent map control parameter as elements in the secret key. The theoretical and experimental study was done on binary sequences generated by tent-map, for a binarization threshold equal to the control parameter. This choice of the binarization threshold (equal to tent map parameter) implied that all typical binary sequences involved in the running-key method comply with the *i.i.d.* model required by the statistical inferences used in this evaluation.

The running-key method was experimentally supported by the usual statistical methods completed with the NIST tests suite. All assessments (theoretical and experimental) provided numerical results that allow an immediate application. These quantitative results consist in providing the range of choice for the control parameter as a function of the number of summations (requested by the application) involved in the running-key method.

The running-key procedure, firstly advanced for the logistic map and considered generally valid for ergodic sources, is here completed by this new exploration on tent map.

As a final remark, the provided generator can be used in cryptographic applications where the pseudo-random binary sequences can be an important component in the enciphering key designing.

We mention that, although the running-key study on tent map was done for a particular choice of the binarization threshold, this method can be extended, with similar results, for other choices of binarization threshold.

REFERENCES

1. A. Vlad, A. Ilyas and A. Luca, Unifying running-key approach and logistic map to generate enciphering sequences, *Annals of Telecommunications,* **vol. 68**, p. 179–186, 2013.
2. A. Vlad, A. Ilyas and A. Luca, A closer view of running-key cipher on natural languages, *Proceedings of the Romanian Academy, Series A*, **vol. 13**, Number 2/2012, pp. 157–166, 2012
3. C. E. Shannon, Prediction and Entropy of Printed English, *Bell Syst. Tech. J.,* **vol. 30**, p. 50–64, 1951.
4. C. E. Shannon, Communication Theory of Secrecy Systems, *Bell Syst. Tech. J.,* **vol. 28**, p. 656–715, 1949.
5. C. E. Shannon, A Mathematical Theory of Communication, *Bell Syst. Tech. J.,* **vol. 27**, p. 379–423, 623–656, 1948.
6. W. Diffie and M. Hellman, Privacy and Authentication: An Introduction in Cryptography, *Proc. IEEE,* **vol. 67**, p. 397–426, 1979.
7. A. Luca, A. Ilyas and A. Vlad, Generating Random Binary Sequences using Tent Map, in Proc. 10th International Symposium on Signals, Circuits and Systems, (ISSCS), Iasi, Romania, pp. 81–84. June 30–July 1, 2011.
8. A. Ilyas, A. Luca and A. Vlad, A study on binary sequences generated by tent map having cryptographic view, in Proc. 9th International Conference on Communications (COMM), Bucharest, pp. 23–26, June 2012.
9. D. Arroyo, G. Alvarez, J. M. Amigo and S. Li, Cryptanalysis of a family of self-syncronizing chaotic stream cipher, Commun Nonlinear Sci Numer Simul **16**, pp. 805–813, 2011.
10. A. Spataru, Fondements de la theorie de la transmission de l'information, Lausanne: Presses Polytechniques Romandes, 1987.
11. J. Devore, Probability and Statistics for Engineering and the Sciences, 2nd ed., Monterey, California: Brooks/Cole Publishing Company, 1987.
12. Runkin *et al.*, Statistical test suite for random and pseudo random number generators for cryptographic applications, *NIST special publication,* **Vols. 800-22**, 2010.

Received May 20, 2013

THE PUBLISHING HOUSE
OF THE ROMANIAN ACADEMY

PROCEEDINGS OF THE ROMANIAN ACADEMY, Series A,
Volume 14, Special Issue 2013, pp. 303–316

TOWARDS AN ALGEBRAIC ATTACK ON AES-128 FASTER THAN BRUTE-FORCE

Andrei SIMION[*], Gabriel NEGARA[**]

[*] Independent researcher, Bucharest, ROMANIA
[**] "Al. I. Cuza" University of Iasi, ROMANIA
Corresponding author: Gabriel NEGARA, E-mail: negara10@gmail.com

In this paper we describe the main ideas of a few versions of an algebraic known plaintext attack against AES-128. The attack could be applied under the hypothesis of knowing a part of the 16-bytes key. These attack versions are based on some specific properties of the key schedule, properties that allow splitting the keys space (2^{128} keys) in subspaces based on some well-defined criteria. The practical efficiency of these attacks depends on some conditions including a conjecture. Also, the paper introduces a definition of weak keys, in the context of the presented attack.

Key words: AES-128, Rijndael, brute-force attack, algebraic attacks, multivariate binary equations system, key schedule, expanded keys.

1. INTRODUCTION

The Advanced Encryption Standard, with its three versions, AES-128, AES-192, and AES-256 is the best known and most widely used block cipher [1]. A detailed description of AES, including some attacks can be found in [2].

In the **2nd Section** of the paper we introduce a scheme, containing the 11 round keys in a compact form which leads to a series of properties of the key schedule, especially the way in which the bytes from the initial key contribute to obtaining the round keys bytes.

Based on these properties, we present some ways in which the 2^{128} keys can be split in classes (subspaces), the computation needed for the expansion of all the keys (at once) within a class being more efficient (reduced) than the computation needed for obtaining the round keys for each key from that specific class.

We also study the case when all the 16 bytes of the initial key K^0 are known, as in the brute-force approaches.

The **3rd Section** presents versions of potential algebraic attacks against AES-128. In these versions 1, 2, 4, or 8 bytes out of 16 of the initial key are considered unknown, while the other 15, 14, 12 or 8 bytes are considered known. It is shown that the 11 round keys can be easily computed even if we consider unknown 8 bytes from the initial key.

The attack versions could be more efficient than a brute-force approach if some conditions are satisfied, including a conjecture.

In this work we present in more detail only the version of the algebraic attack in which just 1 byte from the initial key is considered unknown; for the other possible attack scenarios only the main ideas are presented.

In order to obtain the equations system that needs to be tested for compatibility we use an algorithm for computing S-box's output when providing as input binary vectors.

Improvements of these attacks can be made by taking into account some properties of the S-box and also by using some feasible approaches (like in the case of the key schedule) that could lead to a better structure of the equations systems obtained.

In the **4th Section** we present a definition of keys considered weak in respect to some criteria. For this we expand the key in inverse order, from round 10 to round 0, because in this way is much easier to identify the 'weak keys' in the sense we defined them.

Also, the attack version using the key expanded in inverse order and considering unknown 1 byte of the key K^{10} could be more efficient than the attack considering the key expanded in the usual way and, again, 1 byte of the key unknown.

2. PROPERTIES OF THE KEY SCHEDULE

The initial key K^0 and the 10 round keys K^1, K^2, ..., K^{10} (each containing 16 bytes) are all called "expanded key".

Let us shortly remind the iterative computation of round key K^{i+1} from the round key K^i:

If

$$K^i = \begin{bmatrix} K_{00}^i & K_{10}^i & K_{20}^i & K_{30}^i \\ K_{01}^i & K_{11}^i & \dots & \dots \\ K_{02}^i & \dots & \dots & \dots \\ K_{03}^i & \dots & \dots & \dots \end{bmatrix}$$

then the bytes of the round key K^{i+1} are obtained by:

$$K^{i+1} = \begin{bmatrix} K_{00}^{i+1} = K_{00}^i + SBox(K_{31}^i) + C^i & K_{10}^{i+1} = K_{00}^{i+1} + K_{10}^i & \dots & \dots \\ K_{01}^{i+1} = K_{01}^i + SBox(K_{32}^i) & \dots & \dots & \dots \\ K_{02}^{i+1} = K_{02}^i + SBox(K_{33}^i) & \dots & \dots & \dots \\ K_{03}^{i+1} = K_{03}^i + SBox(K_{30}^i) & \dots & \dots & \dots \end{bmatrix}$$

where + represents the bitwise operation XOR (applied for 2 bytes) and C^i represents the round constant.

It is difficult to notice some properties of the key by describing the expanded key using the above relations.

Using the conventions (for page fitting):

– *xy* meaning *x XOR y*;

– *[x]* meaning *SBox(x);*

and other adequate notations, the expanded key can be written in a compact and easily readable form.

The expanded key is presented in the first 4 columns of Scheme 1, the used notations being written in the 5th column; the values 01, 02, 04, ..., 1B, 36 represent the round constants.

	1	2	3	4	5
0	a	e	i	m	
0	b	f	j	n	
0	c	g	k	o	
0	d	h	l	p	
1	eimA1	imA1	mA1	A1	A1=aeim[n]01
1	fjnB1	jnB1	nB1	B1	B1=bfjn[o]
1	gkoC1	koC1	oC1	C1	C1=cgko[p]
1	hlpD1	lpD1	pD1	D1	D1=dhlp[m]
2	iA1B2	mB2	A1B2	B2	B2=em[B1]02
2	jB1C2	nC2	B1C2	C2	C2=fn[C1]
2	kC1D2	oD2	C1D2	D2	D2=go[D1]
2	lD1A2	pA2	D1A2	A2	A2=hp[A1]
3	mA1B2C3	A1C3	B2C3	C3	C3=im[C2]04
3	nB1C2D3	B1D3	C2D3	D3	D3=jn[D2]
3	oC1D2A3	C1A3	D2A3	A3	A3=ko[A2]
3	pD1A2B3	D1B3	A2B3	B3	B3=lp[B2]

(*continued*)

4	A1B2C3D4	B2D4	C3D4	D4	D4=m[D3]08
4	B1C2D3A4	C2A4	D3A4	A4	A4=n[A3]
4	C1D2A3B4	D2B4	A3B4	B4	B4=o[B3]
4	D1A2B3C4	A2C4	B3C4	C4	C4=p[C3]
5	B2C3D4A5	C3A5	D4A5	A5	A5=A1[A4]10
5	C2D3A4B5	D3B5	A4B5	B5	B5=B1[B4]
5	D2A3B4C5	A3C5	B4C5	C5	C5=C1[C4]
5	A2B3C4D5	B3D5	C4D5	D5	D5=D1[D4]
6	C3D4A5B6	D4B6	A5B6	B6	B6=B2[B5]20
6	D3A4B5C6	A4C6	B5C6	C6	C6=C2[C5]
6	A3B4C5D6	B4D6	C5D6	D6	D6=D2[D5]
6	B3C4D5A6	C4A6	D5A6	A6	A6=A2[A5]
7	D4A5B6C7	A5C7	B6C7	C7	C7=C3[C6]40
7	A4B5C6D7	B5D7	C6D7	D7	D7=D3[D6]
7	B4C5D6A7	C5A7	D6A7	A7	A7=A3[A6]
7	C4D5A6B7	D5B7	A6B7	B7	B7=B3[B6]
8	A5B6C7D8	B6D8	C7D8	D8	D8=D4[D7]80
8	B5C6D7A8	C6A8	D7A8	A8	A8=A4[A7]
8	C5D6A7B8	D6B8	A7B8	B8	B8=B4[B7]
8	D5A6B7C8	A6C8	B7C8	C8	C8=C4[C7]
9	B6C7D8A9	C7A9	D8A9	A9	A9=A5[A8]1B
9	C6D7A8B9	D7B9	A8B9	B9	B9=B5[B8]
9	D6A7B8C9	A7C9	B8C9	C9	C9=C5[C8]
9	A6B7C8D9	B7D9	C8D9	D9	D9=D5[D8]
10	C7D8A9B10	D8B10	A9B10	B10	B10=B6[B9]36
10	D7A8B9C10	A8C10	B9C10	C10	C10=C6[C9]
10	A7B8C9D10	B8D10	C9D10	D10	D10=D6[D9]
10	B7C8D9A10	C8A10	D9A10	A10	A10=A6[A9]

Scheme 1. The expanded key, using some specific notations.

In the scheme above, for example

$$A1=aeim[n]01$$

means

$$A1=a\ XOR\ e\ XOR\ i\ XOR\ m\ XOR\ SBox(n)\ XOR\ 01.$$

Using the notations in column 5 a series of properties of the expanded key can be revealed, especially the way the bytes of the initial key are found as structural parts of the expanded keys.

First, Table 1 indicates the bytes from the initial key K^0 that are components of the expressions of type *A*, *B*, *C* and *D*.

Table 1

Components Bytes in Expressions *A, B, C, D*

	a	*b*	*c*	*d*	*e*	*f*	*g*	*h*	*i*	*j*	*k*	*l*	*m*	*n*	*o*	*p*
A1	*a*				*e*				*i*				*m*	*n*		
A2	*a*				*e*			*h*	*i*				*m*	*n*		*p*
A3-A10	*a*				*e*			*h*	*i*		*k*		*m*	*n*	*o*	*p*
B1		*b*				*f*				*j*				*n*	*o*	
B2		*b*			*e*	*f*				*j*			*m*	*n*	*o*	

Table 1 (continued)

B3-B10		*b*			*e*	*f*				*j*		*l*	*m*	*n*	*o*	*p*
C1			*c*				*g*				*k*				*o*	*p*
C2			*c*			*f*	*g*				*k*			*n*	*o*	*p*
C3-C10			*c*			*f*	*g*		*i*		*k*		*m*	*n*	*o*	*p*
D1				*d*				*h*				*l*	*m*			*p*
D2				*d*			*g*	*h*				*l*	*m*		*o*	*p*
D3-D10				*d*			*g*	*h*		*j*		*l*	*m*	*n*	*o*	*p*

From Table 1 and Scheme 1 one can deduce the properties *P1*, *P2*, *P3*, *P4* and *P5* from Table 2. These properties refer to the number of bytes from the round keys K^1, K^2, ..., K^{10} in which some specific byte or bytes of the initial key K^0 appears.

Table 2

The properties *P1*, *P2*, *P3*, *P4* and *P5*

Property	Bytes from K^0	*K1*	*K2*	*K3*	*K4*	*K5*	*K6*	*K7*	*K8*	*K9*	*K10*
P1	*a*	4	6	8	9	9	9	9	9	9	9
P2	*i*	2	5	8	11	12	12	12	12	12	12
P3	*a, c*	8	12	12	12	12	12	12	12	12	12
P4	*a, i*	4	6	10	12	12	12	12	12	12	12
P5	*a,c,i,k*	8	12	12	12	12	12	12	12	12	12

Let's notice that, due to the symmetry of the expanded key:
– the property *P1* also holds for the bytes *b*, *c* and *d*;
– the property *P2* also holds for the bytes *j*, *k* and *l*;
– the property *P3* also holds for the subset *{b, d}*;
– the property *P4* also holds for the subsets *{b, j}*, *{c, k}* and *{d, l};*
– the property *P5* also holds for the subset *{b, d, j, l}*.
Observation: one can take into consideration the other bytes from the initial key, but their properties are less relevant for our purpose: affecting a number of bytes from the round keys as small as possible.

Property *P6*: the bytes *a, b, c* and *d* pass through the S-Box only individually (never in combination of 2, 3 or 4).

Property *P7*: the bytes *a, b, c, d, i, j, k* and *l* pass through the S-Box only in combination of at most 2 or 3.

We will explain the significance of these properties for our potential attack.

In the following we show that the brute-force approaches could be accelerated, at least in the process of obtaining the expanded keys, and in the 3rd Section we present the main ideas for some versions of an algebraic attack using the enounced properties.

In a naive brute-force attack, the keys are generated and used sequentially; in order to obtain *n* expanded keys, 40 S-Box substitutions and 170 exclusive or operations are performed for *n* times.

As a consequence of the fact that the key schedule process is independent from the enciphering process we can first group the expanded keys in some specific classes and store them using small dimension tables. These tables would contain precomputed bytes or combinations of bytes as components of the expanded keys (for example, from Scheme 1, the byte *A1B2C3B4* can be seen as an exclusive or of 4 components).

There are multiple options for grouping the expanded keys space (2^{128}) in such a way that obtaining a group of n expanded keys is significantly less computationally expensive than the usual way (40 substitution and 170 exclusive or for *n* times).

These possibilities of generating classes of expanded keys depend on the ability to perform operations based on the data presented in Scheme 1.

We present only the main ideas:

– the expressions *A1*, *A2*, ..., *A10* depend only on 9 from the 16 bytes of initial key: *a, e, i, m, n, h, p, k, o*; more than that, these expressions do not explicitly depend on the 9 bytes but only on the combinations *aeim*, *n*, *hp* and *ko*;

– the expressions *A1*, *A2*, ..., *A10*, in combination with the expressions *C1*, *C2*, ..., *C10* depend only on 12 bytes of initial key and, like in the previous case, do not explicitly depend on each byte individually, but only on the combinations *ae*, *im*, *cg, ko, f, h, n* and *p.*

3. ALGEBRAIC ATTACK VERSIONS

All the attack versions we present are variations of the known plaintext brute-force attack. A part of the bytes from the initial key K^0 are considered known, the other bytes being recovered from some binary equations systems. Obviously, like in the case of usual brute-force approach, when the equations system is found to be incompatible, the attack will take into consideration some other values for the bytes considered known.

The practical efficiency of such versions of attacks depends especially on:

– the computational effort needed to generate the equations systems;

– the complexity of solving the system, in case of compatibility;

– the efficiency of the false keys filtering.

Related to the false keys filtering, the following conjecture would represent strong result for the success and efficiency of the attack (if true):

Conjecture: given an over-defined binary equations system that could be compatible or incompatible, it is easier to prove that it is incompatible than to find its solution, when compatible.

This conjecture is true for a binary multivariate equations system, when the number of binary variables is small but, probably, it is hard to decide when the number of variables is larger.

In the following, we present the basic ideas of the attack versions successively considering unknown 1, 2, 4 and 8 bytes from the initial key.

First, we depict the way to obtain the expanded key, starting from the Scheme 1.

The Scheme 2 presents the expanded key obtained when the byte $u = a$ is considered unknown. Similar schemes are obtained when the byte considered unknown is *b*, *c*, or *d*. The expanded key when considering unknown the bytes $u = a$ and $v = c$ is presented in Scheme 3 (similar when the bytes *b* and *d* are unknown). The expanded key in the case of unknown bytes $u = a$, $v = c$, $x = b$ and $y = d$ is presented in Scheme 4.

One can observe that the Scheme 2 can be directly obtained from the Scheme 1:

– the values of the expressions *A1*, *A2*, ..., *A10* from the 5th column are modified appropriately;

– in the columns 1, 2, 3, 4, *A1* becomes *A1P0(u)*, *A2* becomes *A2P1(u)*, *A3* becomes *A3P2(u)*, ..., *A10* becomes *A10P9(u)*.

The way to compute the expressions *P0(u), P1(u)*, *P2(u), ..., P9(u)* is specified above the round key where they appear for the first time.

Similarly, the Scheme 3 can be deduced from the Scheme 2, by modifying the values of the expressions *C1*, *C2*, ..., *C10* from the 5th column and by replacing the expressions *C1*, *C2*, ..., *C10* from the columns 1, 2, 3, 4 with *C1Q0(v)*, *C2Q1(v), ...*, *C10Q9(v).*

Also, the Scheme 4 can be deduced from the Scheme 3.

In the Schemes 2 and 3 one can observe that some of the round keys bytes (in green) do not depend on the bytes considered unknown, in conformity with the properties *P1* and *P3*.

Other possibilities of selecting the unknown bytes are shortly presented, without the corresponding schemes for the expanded keys (these schemes cannot be deduced directly from Scheme 1).

Due to the similarities in the calculus of the expressions of *P, Q, R, S* type we will first present the general methodology of calculus, followed then by the analysis of the particularities specific to each scheme.

One can quickly observe that the expressions like *P1, Q1, R1, S1* can be immediately obtained from the truth tables of *SBox(x+c),* where c is a constant from the set {0, 1, 2, ..., 255}, depending on the bytes values from the initial key, that represent components of the expressions *A1, B1, C1, D1*.

	1	2	3	4	5
0	u	e	i	m	
0	b	f	j	n	
0	c	g	k	o	
0	d	h	l	p	
	P0(u)=u01				
1	eimA1P0(u)	imA1P0(u)	mA1P0(u)	A1P0(u)	A1=eim[n]
1	fjnB1	jnB1	nB1	B1	B1=bfjn[o]
1	gkoC1	koC1	oC1	C1	C1=cgko[p]
1	hlpD1	lpD1	pD1	D1	D1=dhlp[m]
	P1(u)=[A1P0(u)]				
2	iA1B2P0(u)	mB2	A1B2P0(u)	B2	B2=em[B1]02
2	jB1C2	nC2	B1C2	C2	C2=fn[C1]
2	kC1D2	oD2	C1D2	D2	D2=go[D1]
2	lD1A2P1(u)	pA2P1(u)	D1A2P1(u)	A2P1(u)	A2=hp
	P2(u)=[A2P1(u)]				
3	mA1B2C3P0(u)	A1C3P0(u)	B2C3	C3	C3=im[C2]04
3	nB1C2D3	B1D3	C2D3	D3	D3=jn[D2]
3	oC1D2A3P2(u)	C1A3P2(u)	D2A3P2(u)	A3P2(u)	A3=ko
3	pD1A2B3P1(u)	D1B3	A2B3P1(u)	B3	B3=lp[B2]
	P3(u)=[A3P2(u)]				
4	A1B2C3D4P0(u)	B2D4	C3D4	D4	D4=m[D3]08
4	B1C2D3A4P3(u)	C2A4P3(u)	D3A4P3(u)	A4P3(u)	A4=n
4	C1D2A3B4P2(u)	D2B4	A3B4P2(u)	B4	B4=o[B3]
4	D1A2B3C4P1(u)	A2C4P1(u)	B3C4	C4	C4=p[C3]
	P4(u)=[A4P3(u)]P0(u)10				
5	B2C3D4A5P4(u)	C3A5P4(u)	D4A5P4(u)	A5P4(u)	A5=A1
5	C2D3A4B5P3(u)	D3B5	A4B5P3(u)	B5	B5=B1[B4]
5	D2A3B4C5P2(u)	A3C5P2(u)	B4C5	C5	C5=C1[C4]
5	A2B3C4D5P1(u)	B3D5	C4D5	D5	D5=D1[D4]
	P5(u)=[A5P4(u)]P1(u)				
6	C3D4A5B6P4(u)	D4B6	A5B6P4(u)	B6	B6=B2[B5]20
6	D3A4B5C6P3(u)	A4C6P3(u)	B5C6	C6	C6=C2[C5]
6	A3B4C5D6P2(u)	B4D6	C5D6	D6	D6=D2[D5]
6	B3C4D5A6P5(u)	C4A6P5(u)	D5A6P5(u)	A6P5(u)	A6=A2
	P6(u)=[A6P5(u)]P2(u)				
7	D4A5B6C7P4(u)	A5C7P4(u)	B6C7	C7	C7=C3[C6]40
7	A4B5C6D7P3(u)	B5D7	C6D7	D7	D7=D3[D6]
7	B4C5D6A7P6(u)	C5A7P6(u)	D6A7P6(u)	A7P6(u)	A7=A3
7	C4D5A6B7P5(u)	D5B7	A6B7P5(u)	B7	B7=B3[B6]
	P7(u)=[A7P6(u)]P3(u)				
8	A5B6C7D8P4(u)	B6D8	C7D8	D8	D8=D4[D7]80
8	B5C6D7A8P7(u)	C6A8P7(u)	D7A8P7(u)	A8P7(u)	A8=A4
8	C5D6A7B8P6(u)	D6B8	A7B8P6(u)	B8	B8=B4[B7]
8	D5A6B7C8P5(u)	A6C8P5(u)	B7C8	C8	C8=C4[C7]
	P8(u)=[A8P7(u)]P4(u)1B				
9	B6C7D8A9P8(u)	C7A9P8(u)	D8A9P8(u)	A9P8(u)	A9=A5
9	C6D7A8B9P7(u)	D7B9	A8B9P7(u)	B9	B9=B5[B8]
9	D6A7B8C9P6(u)	A7C9P6(u)	B8C9	C9	C9=C5[C8]
9	A6B7C8D9P5(u)	B7D9	C8D9	D9	D9=D5[D8]
	P9(u)=[A9P8(u)]P5(u)				
10	C7D8A9B10P8(u)	D8B10	A9B10P8(u)	B10	B10=B6[B9]36
10	D7A8B9C10P7(u)	A8C10P7(u)	B9C10	C10	C10=C6[C9]
10	A7B8C9D10P6(u)	B8D10	C9D10	D10	D10=D6[D9]
10	B7C8D9A10P9(u)	C8A10P9(u)	D9A10P9(u)	A10P9(u)	A10=A6

Scheme 2. The expanded key - 1 unknown byte of K^o.

	1	2	3	4	5
0	u	e	i	m	
0	b	f	j	n	
0	v	g	k	o	
0	d	h	l	p	
	P0(u)=u01		Q0(v)=v		
1	eimA1P0(u)	imA1P0(u)	mA1P0(u)	A1P0(u)	A1=eim[n]
1	fjnB1	jnB1	nB1	B1	B1=bfjn[o]
1	gkoC1Q0(v)	koC1Q0(v)	oC1Q0(v)	C1Q0(v)	C1=gko[p]
1	hlpD1	lpD1	pD1	D1	D1=dhlp[m]
	P1(u)=[A1P0(u)]		Q1(v)=[C1Q0(v)]		
2	iA1B2P0(u)	mB2	A1B2P0(u)	B2	B2=em[B1]02
2	jB1C2Q1(v)	nC2Q1(v)	B1C2Q1(v)	C2Q1(v)	C2=fn
2	kC1D2Q0(v)	oD2	C1D2Q0(v)	D2	D2=go[D1]
2	lD1A2P1(u)	pA2P1(u)	D1A2P1(u)	A2P1(u)	A2=hp
	P2(u)=[A2P1(u)]		Q2(v)=[C2Q1(v)]04		
3	mA1B2C3P0(u)Q2(v)	A1C3P0(u)Q2(v)	B2C3Q2(v)	C3Q2(v)	C3=im
3	nB1C2D3Q1(v)	B1D3	C2D3Q1(v)	D3	D3=jn[D2]
3	oC1D2A3P2(u)Q0(v)	C1A3P2(u)Q0(v)	D2A3P2(u)	A3P2(u)	A3=ko
3	pD1A2B3P1(u)	D1B3	A2B3P1(u)	B3	B3=lp[B2]
	P3(u)=[A3P2(u)]		Q3(v)=[C3Q2(v)]		
4	A1B2C3D4P0(u)Q2(v)	B2D4	C3D4Q2(v)	D4	D4=m[D3]08
4	B1C2D3A4P3(u)Q1(v)	C2A4P3(u)Q1(v)	D3A4P3(u)	A4P3(u)	A4=n
4	C1D2A3B4P2(u)Q0(v)	D2B4	A3B4P2(u)	B4	B4=o[B3]
4	D1A2B3C4P1(u)Q3(v)	A2C4P1(u)Q3(v)	B3C4Q3(v)	C4Q3(v)	C4=p
	P4(u)=[A4P3(u)]P0(u)10		Q4(v)=[C4Q3(v)]Q0(v)		
5	B2C3D4A5P4(u)Q2(v)	C3A5P4(u)Q2(v)	D4A5P4(u)	A5P4(u)	A5=A1
5	C2D3A4B5P3(u)Q1(v)	D3B5	A4B5P3(u)	B5	B5=B1[B4]
5	D2A3B4C5P2(u)Q4(v)	A3C5P2(u)Q4(v)	B4C5Q4(v)	C5Q4(v)	C5=C1
5	A2B3C4D5P1(u)Q3(v)	B3D5	C4D5Q3(v)	D5	D5=D1[D4]
	P5(u)=[A5P4(u)]P1(u)		Q5(v)=[C5Q4(v)]Q1(v)		
6	C3D4A5B6P4(u)Q2(v)	D4B6	A5B6P4(u)	B6	B6=B2[B5]20
6	D3A4B5C6P3(u)Q5(v)	A4C6P3(u)Q5(v)	B5C6Q5(v)	C6Q5(v)	C6=C2
6	A3B4C5D6P2(u)Q4(v)	B4D6	C5D6Q4(v)	D6	D6=D2[D5]
6	B3C4D5A6P5(u)Q3(v)	C4A6P5(u)Q3(v)	D5A6P5(u)	A6P5(u)	A6=A2
	P6(u)=[A6P5(u)]P2(u)		Q6(v)=[C6Q5(v)]Q2(v)40		
7	D4A5B6C7P4(u)Q6(v)	A5C7P4(u)Q6(v)	B6C7Q6(v)	C7Q6(v)	C7=C3
7	A4B5C6D7P3(u)Q5(v)	B5D7	C6D7Q5(v)	D7	D7=D3[D6]
7	B4C5D6A7P6(u)Q4(v)	C5A7P6(u)Q4(v)	D6A7P6(u)	A7P6(u)	A7=A3
7	C4D5A6B7P5(u)Q3(v)	D5B7	A6B7P5(u)	B7	B7=B3[B6]
	P7(u)=[A7P6(u)]P3(u)		Q7(v)=[C7Q6(v)]Q3(v)		
8	A5B6C7D8P4(u)Q6(v)	B6D8	C7D8Q6(v)	D8	D8=D4[D7]80
8	B5C6D7A8P7(u)Q5(v)	C6A8P7(u)Q5(v)	D7A8P7(u)	A8P7(u)	A8=A4
8	C5D6A7B8P6(u)Q4(v)	D6B8	A7B8P6(u)	B8	B8=B4[B7]
8	D5A6B7C8P5(u)Q7(v)	A6C8P5(u)Q7(v)	B7C8Q7(v)	C8Q7(v)	C8=C4
	P8(u)=[A8P7(u)]P4(u)1B		Q8(v)=[C8Q7(v)]Q4(v)		
9	B6C7D8A9P8(u)Q6(v)	C7A9P8(u)Q6(v)	D8A9P8(u)	A9P8(u)	A9=A5
9	C6D7A8B9P7(u)Q5(v)	D7B9	A8B9P7(u)	B9	B9=B5[B8]
9	D6A7B8C9P6(u)Q8(v)	A7C9P6(u)Q8(v)	B8C9Q8(v)	C9Q8(v)	C9=C5
9	A6B7C8D9P5(u)Q7(v)	B7D9	C8D9Q7(v)	D9	D9=D5[D8]
	P9(u)=[A9P8(u)]P5(u)		Q9(v)=[C9Q8(v)]Q5(v)		
10	C7D8A9B10P8(u)Q6(v)	D8B10	A9B10P8(u)	B10	B10=B6[B9]36
10	D7A8B9C10P7(u)Q9(v)	A8C10P7(u)Q9(v)	B9C10Q9(v)	C10Q9(v)	C10=C6
10	A7B8C9D10P6(u)Q8(v)	B8D10	C9D10Q8(v)	D10	D10=D6[D9]
10	B7C8D9A10P9(u)Q7(v)	C8A10P9(u)Q7(v)	D9A10P9(u)	A10P9(u)	A10=A6

Scheme 3. The expanded key - 2 unknown bytes of K^o.

	1	2
0	u	e
0	x	f
0	v	g
0	y	h
	P0(u)=u01	Q0(v)=v
1	eimA1P0(u)	imA1P0(u)
1	fjnB1R0(x)	jnB1R0(x)
1	gkoC1Q0(v)	koC1Q0(v)
1	hlpD1S0(y)	lpD1S0(y)
	P1(u)=[A1P0(u)]	Q1(v)=[C1Q0(v)]
2	iA1B2P0(u)R1(x)	mB2R1(x)
2	jB1C2Q1(v)R0(x)	nC2Q1(v)
2	kC1D2Q0(v)S1(y)	oD2S1(y)
2	lD1A2P1(u)S0(y)	pA2P1(u)
	P2(u)=[A2P1(u)]	Q2(v)=[C2Q1(v)]04
3	mA1B2C3P0(u)Q2(v)R1(x)	A1C3P0(u)Q2(v)
3	nB1C2D3Q1(v)R0(x)S2(y)	B1D3R0(x)S2(y)
3	oC1D2A3P2(u)Q0(v)S1(y)	C1A3P2(u)Q0(v)
3	pD1A2B3P1(u)R2(x)S0(y)	D1B3R2(x)S0(y)
	P3(u)=[A3P2(u)]	Q3(v)=[C3Q2(v)]
4	A1B2C3D4P0(u)Q2(v)R1(x)S3(y)	B2D4R1(x)S3(y)
4	B1C2D3A4P3(u)Q1(v)R0(x)S2(y)	C2A4P3(u)Q1(v)
4	C1D2A3B4P2(u)Q0(v)R3(x)S1(y)	D2B4R3(x)S1(y)
4	D1A2B3C4P1(u)Q3(v)R2(x)S0(y)	A2C4P1(u)Q3(v)
	P4(u)=[A4P3(u)]P0(u)10	Q4(v)=[C4Q3(v)]Q0(v)
5	B2C3D4A5P4(u)Q2(v)R1(x)S3(y)	C3A5P4(u)Q2(v)
5	C2D3A4B5P3(u)Q1(v)R4(x)S2(y)	D3B5R4(x)S2(y)
5	D2A3B4C5P2(u)Q4(v)R3(x)S1(y)	A3C5P2(u)Q4(v)
5	A2B3C4D5P1(u)Q3(v)R2(x)S4(y)	B3R2(x)D5S4(y)
	P5(u)=[A5P4(u)]P1(u)	Q5(v)=[C5Q4(v)]Q1(v)
6	C3D4A5B6P4(u)Q2(v)R5(x)S3(y)	D4B6R5(x)S3(y)
6	D3A4B5C6P3(u)Q5(v)R4(x)S2(y)	A4C6P3(u)Q5(v)
6	A3B4C5D6P2(u)Q4(v)R3(x)S5(y)	B4D6R3(x)S5(y)
6	B3C4D5A6P5(u)Q3(v)R2(x)S4(y)	C4A6P5(u)Q3(v)
	P6(u)=[A6P5(u)]P2(u)	Q6(v)=[C6Q5(v)]Q2(v)40
7	D4A5B6C7P4(u)Q6(v)R5(x)S3(y)	A5C7P4(u)Q6(v)
7	A4B5C6D7P3(u)Q5(v)R4(x)S6(y)	B5D7R4(x)S6(y)
7	B4C5D6A7P6(u)Q4(v)R3(x)S5(y)	C5A7P6(u)Q4(v)
7	C4D5A6B7P5(u)Q3(v)R6(x)S4(y)	D5B7R6(x)S4(y)
	P7(u)=[A7P6(u)]P3(u)	Q7(v)=[C7Q6(v)]Q3(v)
8	A5B6C7D8P4(u)Q6(v)R5(x)S7(y)	B6D8R5(x)S7(y)
8	B5C6D7A8P7(u)Q5(v)R4(x)S6(y)	C6A8P7(u)Q5(v)
8	C5D6A7B8P6(u)Q4(v)R7(x)S5(y)	D6B8R7(x)S5(y)
8	D5A6B7C8P5(u)Q7(v)R6(x)S4(y)	A6C8P5(u)Q7(v)
	P8(u)=[A8P7(u)]P4(u)1B	Q8(v)=[C8Q7(v)]Q4(v)
9	B6C7D8A9P8(u)Q6(v)R5(x)S7(y)	C7A9P8(u)Q6(v)
9	C6D7A8B9P7(u)Q5(v)R8(x)S6(y)	D7B9R8(x)S6(y)
9	D6A7B8C9P6(u)Q8(v)R7(x)S5(y)	A7C9P6(u)Q8(v)
9	A6B7C8D9P5(u)Q7(v)R6(x)S8(y)	B7D9R6(x)S8(y)
	P9(u)=[A9P8(u)]P5(u)	Q9(v)=[C9Q8(v)]Q5(v)
10	C7D8A9B10P8(u)Q6(v)R9(x)S7(y)	D8B10R9(x)S7(y)
10	D7A8B9C10P7(u)Q9(v)R8(x)S6(y)	A8C10P7(u)Q9(v)
10	A7B8C9D10P6(u)Q8(v)R7(x)S9(y)	B8D10R7(x)S9(y)
10	B7C8D9A10P9(u)Q7(v)R6(x)S8(y)	C8A10P9(u)Q7(v)

Scheme 4 (1/2). The expanded key - 4 unknown bytes of K^o (the first 2 columns).

	3	4	5
0	i	m	
0	j	n	
0	k	o	
0	l	p	
	R0(x)=x	S0(y)=y	
1	mA1P0(u)	A1P0(u)	A1=eim[n]
1	nB1R0(x)	B1R0(x)	B1=bfjn[o]
1	oC1Q0(v)	C1Q0(v)	C1=gko[p]
1	pD1S0(y)	D1S0(y)	D1=hlp[m]
	R1(x)=[B1R0(x)]02	S1(y)=[D1S0(y)]	
2	A1B2P0(u)R1(x)	B2R1(x)	B2=em
2	B1C2Q1(v)R0(x)	C2Q1(v)	C2=fn
2	C1D2Q0(v)S1(y)	D2S1(y)	D2=go
2	D1A2P1(u)S0(y)	A2P1(u)	A2=hp
	R2(x)=[B2R1(x)]	S2(y)=[D2S1(y)]	
3	B2C3Q2(v)R1(x)	C3Q2(v)	C3=im
3	C2D3Q1(v)S2(y)	D3S2(y)	D3=jn
3	D2A3P2(u)S1(y)	A3P2(u)	A3=ko
3	A2B3P1(u)R2(x)	B3R2(x)	B3=lp
	R3(x)=[B3R2(x)]	S3(y)=[D3S2(y)]08	
4	C3D4Q2(v)S3(y)	D4S3(y)	D4=m
4	D3A4P3(u)S2(y)	A4P3(u)	A4=n
4	A3B4P2(u)R3(x)	B4R3(x)	B4=o
4	B3C4Q3(v)R2(x)	C4Q3(v)	C4=p
	R4(x)=[B4R3(x)]R0(x)	S4(S0(y))=[D4S3(y)]S0(y)	
5	D4A5P4(u)S3(y)	A5P4(u)	A5=A1
5	A4B5P3(u)R4(x)	B5R4(x)	B5=B1
5	B4C5Q4(v)R3(x)	C5Q4(v)	C5=C1
5	C4D5Q3(v)S4(y)	D5S4(y)	D5=D1
	R5(x)=[B5R4(x)]R1(x)20	S5(y)=[D5S4(y)]S1(y)	
6	A5B6P4(u)R5(x)	B6R5(x)	B6=B2
6	B5C6Q5(v)R4(x)	C6Q5(v)	C6=C2
6	C5D6Q4(v)S5(y)	D6S5(y)	D6=D2
6	D5A6P5(u)S4(y)	A6P5(u)	A6=A2
	R6(x)=[B6R5(x)]R2(x)	S6(y)=[D6S5(y)]S2(y)	
7	B6C7Q6(v)R5(x)	C7Q6(v)	C7=C3
7	C6D7Q5(v)S6(y)	D7S6(y)	D7=D3
7	D6A7P6(u)S5(y)	A7P6(u)	A7=A3
7	A6B7P5(u)R6(x)	B7R6(x)	B7=B3
	R7(x)=[B7R6(x)]R3(x)	S7(y)=[D7S6(y)]S3(y)80	
8	C7D8Q6(v)S7(y)	D8S7(y)	D8=D4
8	D7A8P7(u)S6(y)	A8P7(u)	A8=A4
8	A7B8P6(u)R7(x)	B8R7(x)	B8=B4
8	B7C8Q7(v)R6(x)	C8Q7(v)	C8=C4
	R8(x)=[B8R7(x)]R4(x)	S8(y)=[D8S7(y)]S4(y)	
9	D8A9P8(u)S7(y)	A9P8(u)	A9=A5
9	A8B9P7(u)R8(x)	B9R8(x)	B9=B5
9	B8C9Q8(v)R7(x)	C9Q8(v)	C9=C5
9	C8D9Q7(v)S8(y)	D9S8(y)	D9=D5
	R9(x)=[B9R8(x)]R5(x)36	S9(y)=[D9S8(y)]S5(y)	
10	A9B10P8(u)R9(x)	B10R9(x)	B10=B6
10	B9C10Q9(v)R8(x)	C10Q9(v)	C10=C6
10	C9D10Q8(v)S9(y)	D10S9(y)	D10=D6
10	D9A10P9(u)S8(y)	A10P9(u)	A10=A6

Scheme 4 (2/2). The expanded key - 4 unknown bytes of K^o (the last 3 columns).

In order to compute the values of the other expressions, the output of the S-Box need to be computed first, when providing also vector expressions as input.

Thus, in the key schedule process as well as in the round enciphering process, computing expressions of the form *Sbox(c + P)* is needed, where *c* is a byte constant, and *P* is formed by 8 vectors. These 8 vectors are of dimension 2^{8k}, where k ∈{1, 2, 4} represents the number of bytes in the K^0 key, and can be expressed using a truth table or an Algebraic Normal Form table.

In order to perform such computation, we successively found **3 different methods**. The 3rd method is at least two times faster than the 2nd method and thousands times faster than the 1st one. This 3rd method uses only the truth tables of *P* and *SBox*.

So, we only need to compute *c+ P* and then to compose the functions *SBox* and *(c+ P)*.

Note. We present the first two methods of computing *Sbox(c + P)* in **Appendix**, as some steps of them could be interesting by themselves (including an algorithm for multiplying multivariate binary polynomials).

Regarding the effective calculus of the expressions of type *P, Q, R, S* we consider that for each scheme out of 3 (2, 3 and 4) one can find precomputation options, at least for a part of these expressions. This can be done by using ideas similar to the ones presented in the end of Section 2.

Also, in the same register, one can precompute other components of the expanded key, depending on the known (considered) bytes from the initial key.

Related to the attack versions in which we consider unknown other combinations of bytes from the initial key than the ones in the Schemes 2, 3 and 4, we can mention that these versions could present some advantages and also disadvantages. For example, considering as unknown the bytes *(a, i, c, k)* or *(b, j, d, l)*, the expanded key contains also some bytes that do not depend on the unknown bytes (due to the *P5* property). Meanwhile, the expanded key obtained in the case of considering as unknown the bytes, *a, b, c, d* do not contain such bytes. But, the unknown bytes *a, b, c, d* do not pass simultaneously through the *S-Box*, computing the involved expressions implying only 2^8-length arrays, while the unknown bytes *a, i, c, k* pass simultaneously through the *S-Box (a* with *i*, *c* with *k)*, for computing the involved expressions being necessary arrays of dimension 2^{16}.

Regarding the algebraic attack version involving 8 unknown bytes *(a, b, c, d, i, j, k, l)*, this could be the most efficient one; the vectorial expressions depend only on combinations of 3 out of 8 unknown values: *{ai, k}*, *{bj, l}*, *{ck, i}* and *{dl, j}*.

Thus, these polynomial expressions could be computed using polynomial expressions of length 2^{24}.

Obtaining the equations systems can be done relatively easily when considering only 1 unknown byte. One could use either the direct key expansion from the Scheme 2, or the inverse expansion from the Scheme 5. In these cases, the length of the binary arrays remains at 256 after passing through the S-Box in the enciphering process. For the *MixColumns* operation, one can use the 32 × 32 binary equivalent matrix; for decreasing the number of operations required, it is recommended for the equations systems computation algorithm to use a meet-in-the-middle strategy.

For the attack versions involving more unknown bytes, the difficulties in computing the equations systems increase due to the simultaneous passing through the S-Box of the unknown bytes, during the enciphering process. When using the vectorial expressions of the multivariate polynomials, the lengths of these vectors could reach infeasible length. For example, for 8 unknown bytes, the length of these vectors could have 2^{64} length.

As a consequence, for these scenarios, new strategies must be used in order to efficiently compute the equations systems, for example by introducing additional variables.

4. INVERSE KEY EXPANSION AND WEAK KEYS

In order to obtain the round key K^i from K^{i-1} the following operations need to be performed (from right to the left):

$$\begin{aligned}
&K_{00}^{i-1} = K_{00}^{i} + SBox(K_{31}^{i-1}) + C^{i-1} \qquad K_{10}^{i-1} = K_{10}^{i} + K_{00}^{i} \qquad \cdots\cdots \qquad K_{30}^{i-1} = K_{30}^{i} + K_{20}^{i}\\
&K_{01}^{i-1} = K_{01}^{i} + SBox(K_{32}^{i-1}) \qquad \cdots \qquad \cdots\cdots \qquad \cdots\\
&K_{02}^{i-1} = K_{02}^{i} + SBox(K_{33}^{i-1}) \qquad \cdots \qquad \cdots\cdots \qquad \cdots\\
&K_{03}^{i-1} = K_{03}^{i} + SBox(K_{30}^{i-1}) \qquad K_{13}^{i-1} = K_{13}^{i} + K_{03}^{i} \qquad \cdots\cdots \qquad K_{33}^{i-1} = K_{33}^{i} + K_{23}^{i}
\end{aligned}$$

	1	2	3	4
0	xr	r	xr	r
0	P3(x)r	P3(x)r	r	r
0	[P1(x)Q1(x)r5]P2(x)r	r	r	r
0	P1(x)Q1(x)r	P1(x)Q1(x)r	P1(x)Q1(x)r	P1(x)Q1(x)r5
1	xr	xr	r	r
1	P3(x)r	r	r	r
1	P2(x)r	P2(x)r	P2(x)r	P2(x)r4
1	P1(x)Q1(x)r	r	P1(x)Q1(x)r	r
	P3(x)=[P2(x)r4]			
2	xr	r	r	r
2	r	r	r	r
2	P2(x)r	R	P2(x)r	r
2	P1(x)Q1(x)r	P1(x)Q1(x)r	r	r
3	xr	xr	xr	xr3
3	r	r	r	r
3	P2(x)r	P2(x)r	r	r
3	P1(x)Q1(x)r	r	r	r
	Q1(x)=[xr3]			
4	xr	r	xr	r
4	r	r	r	r
4	P2(x)r	r	r	r
4	P1(x)r	P1(x)r	P1(x)r	P1(x)r2
	P2(x)=[P1(x)r2]			
5	xr	xr	r	r
5	r	r	r	r
5	r	r	r	r
5	P1(x)r	r	P1(x)r	r
6	xr	r	r	r
6	r	r	r	r
6	r	r	r	r
6	P1(x)r	P1(x)r	r	r
7	xr	xr	xr	xr1
7	r	r	r	r
7	r	r	r	r
7	P1(x)r	r	r	r
	P1(x)=[xr1]			
8	xr	r	xr	r
8	r	r	r	r
8	r	r	r	r
8	r	r	r	r
9	xr	xr	r	r
9	r	r	r	r
9	r	r	r	r
9	r	r	r	r
10	a=x	e=r	i=r	m=r
10	b=r	f=r	j=r	n=r
10	c=r	g=r	k=r	o=r
10	d=r	h=r	l=r	p=r
	r1=eim	r2=hp	r3=eim[n[ko]]40	

Scheme 5. The inverse key expansion - 1 unknown bytes of K^{10}.

For this inverse expansion, from K^{10} to K^0, we did not find yet a compact scheme similar to the direct expansion from Scheme 1. So, in Scheme 5 we present a simplified version, and only for the case in which the only byte considered unknown is K_{00}^{10}.

The unknown byte is called *x* and all the other bytes will be identified by *r*, even their values are not equal. The propagation of the unknown byte was studied and was taken into consideration the fact that the xor operations between the known bytes lead also to known bytes; the same for the S-Box operations.

The naming conventions are the same as in the previous schemes.

One can observe that in Scheme 5 there are 122 known bytes of the expanded key and only 54 unknown bytes, in contrast with the Scheme 1, in which we have only 94 bytes considered known and 82 unknown.

Thus, from the point of view on an algebraic attach version, the Scheme 5 would be more feasible. For this case we did not analyze techniques for precomputation. The Scheme 5 presents another interesting property that leads to a definition of weak keys.

Definition. A key is "weak" in relation with an unknown-considered byte (or a group of bytes) if, for a *relatively large* set of values of the known bytes, the corresponding expanded keys contains less unknown bytes than in the *usual* case.

In our scenario, having the K_{00}^{10} byte unknown, we may consider weak those keys for which *r1 = r3*, because in this case $P_1(x)=Q_1(x)$, determining another 9 bytes of the expanded keys to become known and one more byte, K_{02}^{0}, to have a simpler expression.

After performing a series of operations, we got the relations *r1=eim* and *r3=eim[n[ko]]40*, thus one can consider weak keys those that satisfy the relation *[n[ko]]=40*, with *n, k, o* bytes from the key K^{10} and 40 the constant of round 7.

Similarly, one can find weak keys in relation with K_{01}^{10}, K_{02}^{10} or K_{03}^{10}.

CONCLUSIONS

In this work we present a series of properties of the expanded key; based on these properties and using adequate notations, we obtain several schemes of the expanded key, when considering as known all or part of the bytes from the initial key, while the remaining bytes are considered unknown. The schemes are simple and provide a way to split the expanded keys space in some specific classes. A significant part of these classes' components can be precomputed and stored in tables having reasonable dimensions.

Also, related to the inverse key schedule, we introduced a definition of weak keys.

Using the 3rd method for computing expressions of type *Sbox(c + P)*, in combination with precomputation techniques and an efficient implementation of the *MixColumn* operation (adapted to our approach), we are confident in the existence of an attack on AES-128 faster than naive brute-force.

In consequence, we consider that our study is likely to contribute to the design of such a new attack against AES-128.

FUTURE RESEARCH DIRECTIONS

1. The study of the key schedule, using our approach, for AES-192 and AES-256, looking for some additional properties, like the ones used in the related-key attacks known in the literature [3], [4].

2. The design of more precomputation techniques, for the known bytes and for the polynomials expressions depending on the unknown bytes.

3. The identification of some properties of binary multivariate equations systems for the AES-128 case study that may lead to more efficient filtering of the false keys.

4. The design of a new hybrid attack against AES based on the ideas from this paper, from [5] and using the notion of higher order correlations [6].

APPENDIX

In the first two methods of computing *Sbox(c + P)* we use the well-known algorithm of computing the *Algebraic Normal Form (ANF)* of a Boolean function from the truth table (or vice versa). We name this algorithm „*alg0*".

For a Boolean function

$$f:\{0,1\}^n \rightarrow \{0,1\}^m,$$

if f_T represents the function's truth table, and f_A represents the table of vectors corresponding to the m *ANF* function's components (both tables having dimension $2^n \times m$), it is known that

$$alg0(f_T) = f_A \text{ and } alg0(f_A)=f_T.$$

The 1st method of computing Sbox(c + P).

For this method we use only the *ANF* of the *AES SBox*; *P* represents the 8 vectors corresponding to the 8 polynomials, each having n binary variables.

The $c+P$ operations modifies only the components of the free terms of the 8 vectors. Let $Q=c+P$. In order to perform the operations, we need to replace the variables $x_0, x_1, ..., x_7$ from the *ANF* expressions of the *SBox* by the vectors $Q_0, Q_1, ..., Q_7$. Thus, we need to compute all the „monomials of polynomials" and then to sum them based on the active bits from the *8 ANF* expressions of the *SBox*.

First, we need to perform multiplication operations for some multivariate polynomials of n binary variables.

Let R_{A1} and R_{A2}, respectively R_{T1} and R_{T2}, be the vectors of two multivariate polynomials of n binary variables, corresponding to the *ANF* representations, and respectively, to the truth tables.

We have

$$(1)\ R_{A1} \cdot R_{A2} = alg0(R_{T1}) \cdot alg0(R_{T2}) = alg0(R_{T1} \cdot R_{T2}) = alg0(alg0(R_{A1}) \cdot alg0(R_{A2})).$$

Thus, the multiplication of two multivariate polynomials reduces to two applications of the *alg0* algorithm and the multiplication of the corresponding truth tables (bitwise AND).

This algorithm can be easily extended for the multiplication of *m* multivariate polynomials of *n* binary variables.

The number of operations involved can be reduced by using a $2^k \times 2^k$ table containing the precomputed multiplications of multivariate polynomials of *k* binary variables.

Note*:* this multiplication algorithm was developed by the authors in 2012. Recently, we found a very similar algorithm, published in April, 2013 [7]. The authors claim that their algorithm, „*MultANF*", is new and faster than other algorithms in the literature, for example the one in the software package SAGE (www.sagemath.org).

Because this first method involves a relatively large number of operations (247 multiplications of binary multivariate polynomials, then XOR operations based on the active bits from the 8 *ANF* expressions of the *SBox*) we looked for another method to compute the „monomials of polynomials".

The 2nd method of computing Sbox(c + P).

We first present a more general case. Let f and g be two boolean functions:

$$f:\{0,1\}^p \rightarrow \{0,1\}^n,\ g:\{0,1\}^n \rightarrow \{0,1\}^m$$

and

$$h=g \circ f,\ h:\{0,1\}^p \rightarrow \{0,1\}^m.$$

Let f_A, g_A, h_A, and f_T, g_T, h_T, be the ANF tables and, respectively, the truth tables.

We have $h_A=g_A \circ f_A$ and $h_T=g_T \circ f_T$, resulting that:

$$(2)\ h_A = alg0(h_T) = alg0(g_T \circ f_T) = alg0(alg0(g_A) \circ alg0(f_A)).$$

Thus, the calculus of h_A starting from f_A and g_A reduces to the computation of f_T and h_T using *alg0*, the composition of the two functions in order to obtain h_T and finally to a new application of *alg0* for obtaining h_A.

In order to use this method for computing *SBox(c+P)* in our case, we set:

$$f = c + P,\ g = SBox,\ n = m = 8,\ p = 8k,$$

where $k \in \{1, 2, 4\}$ represents the number of unknown key bytes.

REFERENCES

1. Joan Daemen, Vincent Rijmen, *The Design of Rijndael. AES - The Advanced Encryption Standard,* Springer-Verlag, 2002.
2. Lars R. Knudsen, Matthew J. B. Robshaw, *The Block Cipher Companion,* Springer-Verlag, 2011.
3. Alex Biryukov, Dmitry Khovratovich, *Related-keyCryptanalysis of the full AES-192 and AES-256,* 2009.
4. Alex Biryukov, Orr Dunkelman, Natahn Keller, Dmitry Khovratovich, Adi Shamir, *Key Recovery Attacks of Practical Complexity on AES Variants With Up To 10 Rounds*, EUROCRYPT, 2010.
5. Nicolas T. Courtois, Josef Pieprzyk, *Cryptnalysis of Block Ciphers with Overdefined Systems of Equations*, eprint.iacr.org/2002/044.pdf, 2002.
6. Andrei Simion, *Contributions to Cryptanalysis of LFSR-based Stream Ciphers*, PhD Thesis, University of Bucharest, Romania, 2008.
7. Subhabrata Samajder, Palash Sarkar, *Fast multiplication of the algebraic normal forms of two Boolean functions,* WCC 2013 - International Workshop on Coding and Cryptography, Bergen, Norway, 2013.

Received June 1, 2013

THE PUBLISHING HOUSE
OF THE ROMANIAN ACADEMY

PROCEEDINGS OF THE ROMANIAN ACADEMY, Series A,
Volume 14, Special Issue 2013, pp. 317–327

UNVEILLING THE PASSWORD ENCRYPTION PROCESS UNDER WINDOWS – A PRACTICAL ATTACK

Lucian OPREA*

*Independent researcher, Bucharest, ROMANIA
Corresponding author: Lucian OPREA, E-mail: luc.oprea@gmail.com

In this paper we present a practical attack that is intended to break the encryption process used to protect the users' passwords in some versions of the Windows operating system.

Key words: password encryption, Windows, brute-force attack, password hashes.

1. INTRODUCTION

Local and domain account information, such as user passwords, group definitions, and domain associations are stored by the Windows operating system in registry. By default, the registry keys like *HKEY_LOCAL_MACHINE\SAM*, which holds most of this information, are unreadable even for the system administrator account. *SAM* stands for the Security Accounts Manager and is essentially a database of security information, user permissions and passwords. It is sometimes referred to as the Windows local security database.

In the **2nd section** of this paper we present the structure of the *SAM* registry and the location of the values we need in order to launch the attack.

In the **3rd section** we present the cryptographic transformations applied to the user password. We grouped these transformations in three different "encryption" levels. These levels are applied sequentially, the resulted values being stored into the registry structure. We call these levels "encryption levels" due to the cryptographic processing of the initial and intermediate values and even if these levels contain hash primitives.

In order to recover the password of a Windows account, we implemented 3 different applications. A Windows service and a standalone application were implemented in order to gather all the data we need for the attack and to ease the final step of finding the password. The 3rd application, which recovers the password by brute-force, is implemented in CUDA technology in order to use the processing power of a compatible GPU card.

The **4th section** presents the description of the attack and a practical example of recovering an Administrator password.

Observation: the techniques shown here are strictly for educational purposes and should not be used against systems for which you do not have authorization for.

2. SAM STRUCTURE

The initial information we need in order to launch the attack is stored in Windows registry. The most important registry key in our attack is *HKEY_LOCAL_MACHINE\SAM\SAM\Domains*. This key contains two other subkeys:

– *Account* subkey: contains information about defined user and group accounts. *Administrator* and *Guest* information is stored here, although these accounts are defined by default;

– *BuiltIn* subkey: contains information about operating system users defined by default.

HKEY_LOCAL_MACHINE\SAM\SAM\Domains\Account\Names contains the list of all user accounts on the machine. Every user account in here has a key containing a hex value which is the *RID* (Relative Identifier) of the account or group.

In *HKEY_LOCAL_MACHINE\SAM\SAM\Domains\Account\Users* each account is defined by a different subkey which is the *RID* (Relative Identifier) of that account or group. For the *Administrator* account, that is the account against which we apply the example attack, the *RID* value is the constant value *0x000001F4* and it is not dependent on the computer.

For every user account, this key contains two *REG_BINARY* values (*F* and *V*) which contains some of the data we need [1].

The following information can be extracted from the *V* value:

Table 1

User information stored in *HKEY_LOCAL_MACHINE\SAM\SAM\Domains\Account\Users\{RID}\V*

Address	Information
0x0c	Offset of the account name
0x10	Length of the account name
0x18	Offset of the complete account name
0x1c	Length of the complete account name
0x24	Offset of the comment
0x28	Length of the comment
0x48	Offset of the homedir name
0x4c	Length of the homedir name
0x9c	*Offset of the final password hashes (SAM)*
0xc4	Number of hashes from history

In order to extract a value, we must add *0xcc* to the offset value from the table above.

For example, the final hashes offset is computed as being *V[0x9c] + 0xcc,* the first one starting at the computed offset being *LMHash*, followed by *NTHash*. If *SYSKEY* is enabled, the encrypted hashes are prefixed by *0x10000000* [2].

3. ENCRYPTION LEVELS

The process of encrypting the user password is done in three different steps, on three different levels of encryption. The figure below presents the notations used in this document for referring the output of each of these levels.

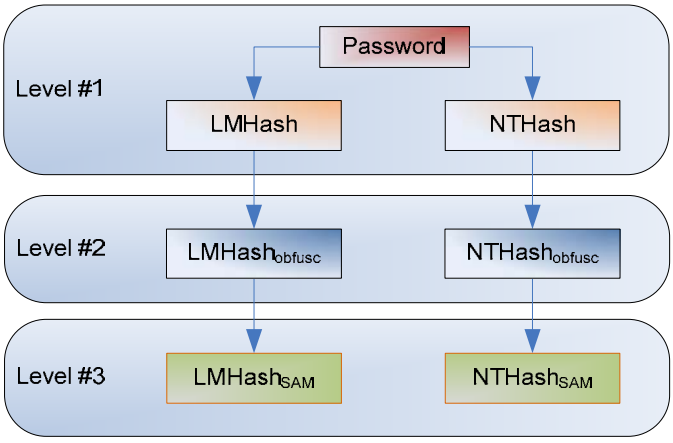


Fig. 1 – Encryption levels.

The *V* registry value presented in the previous section contains the two final hashes for the password of the corresponding user account:

– $LMHash_{SAM}$: LAN Manager hash, was the primary hash that Microsoft LAN Manager and Microsoft Windows versions prior to Windows NT (MS Client for DOS, Windows for Workgroups, Windows 95/98/Me and in some cases Windows NT or newer) used to store user passwords. Support for the legacy LAN Manager protocol continued in later versions of Windows for backward compatibility, but was recommended by Microsoft to be turned off by administrators; as of Windows Vista, the protocol is disabled by default [7];

– $NTHash_{SAM}$: used in Windows NT/2000/2003/XP.

The first encryption level is presented in the picture below.

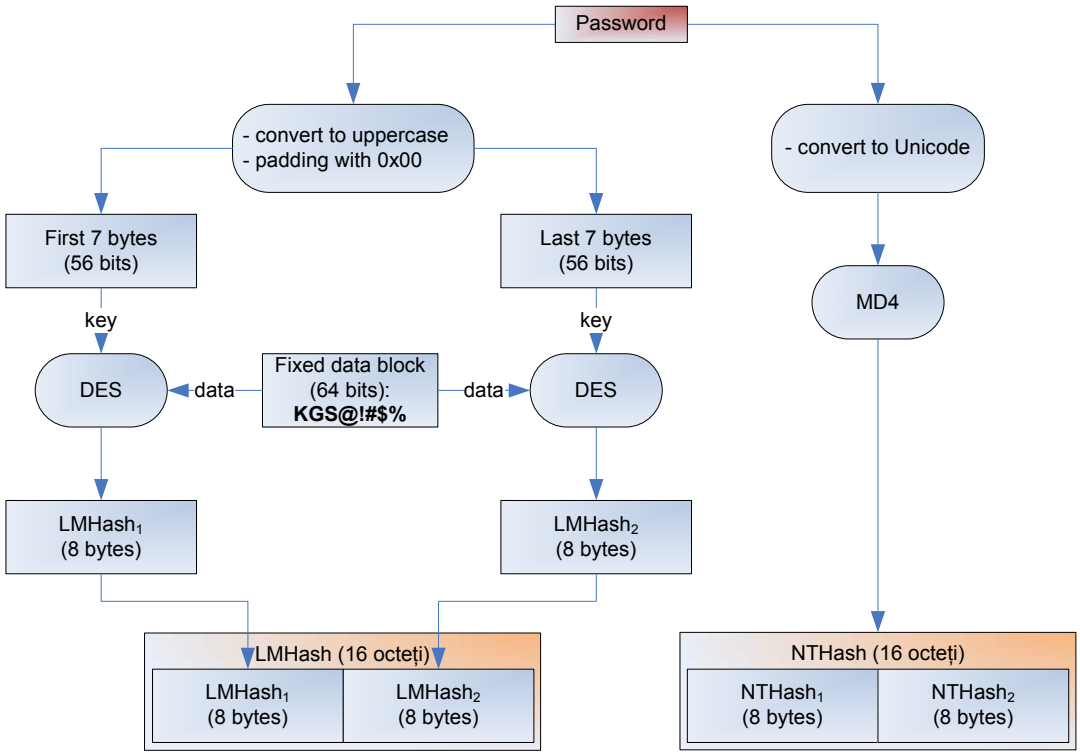


Fig. 2 – The 1st encryption level.

One could observe the following vulnerabilities in *LMHash* computation:

– the password is first converted to uppercase, thus it reduces the number of possible characters;

– the password is used to encrypt a known fixed plaintext, making it susceptible to known plaintext cryptanalysis;

– no salt, IV or any sort of randomness; two identical passwords will generate two identical hashes after in the first level;

– password padding is done with the byte *0x00* to a length of 14 characters. So, a password with at most 7 characters will have *0x00* on all its 7 MSBs; the ciphertext resulted from this fixed block is known so it is easy to find if a password is less than 8 characters;

– the password is split in two parts which then are used independently; in this way, one could attack the two parts separately.

To avoid the *LMHash* vulnerabilities, there are some actions than could be performed:

– use a password with more than 14 characters. The algorithm for *LMHash* generation supports only passwords less than this length, so it will not be generated;

– disable *LMHash* generation (Windows 2000/XP/2003):

- *HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Control\Lsa\NoLmHash: REG_DWORD=1*, or:

- using GPO (Network Security: Do not store LAN Manager hash value on the next password change).

The drawback of disabling the generation of the *LMHash* is that if for an user account this hash is not present, that user cannot authenticate from computers which support only the LM authentication protocol (Windows 95/98 or older).

To add another layer of protection to the hashes provided by the 1st encryption level, Windows adds the 2nd encryption layer, using *DES* and keys derived from the user's *RID*, as presented in the picture below.

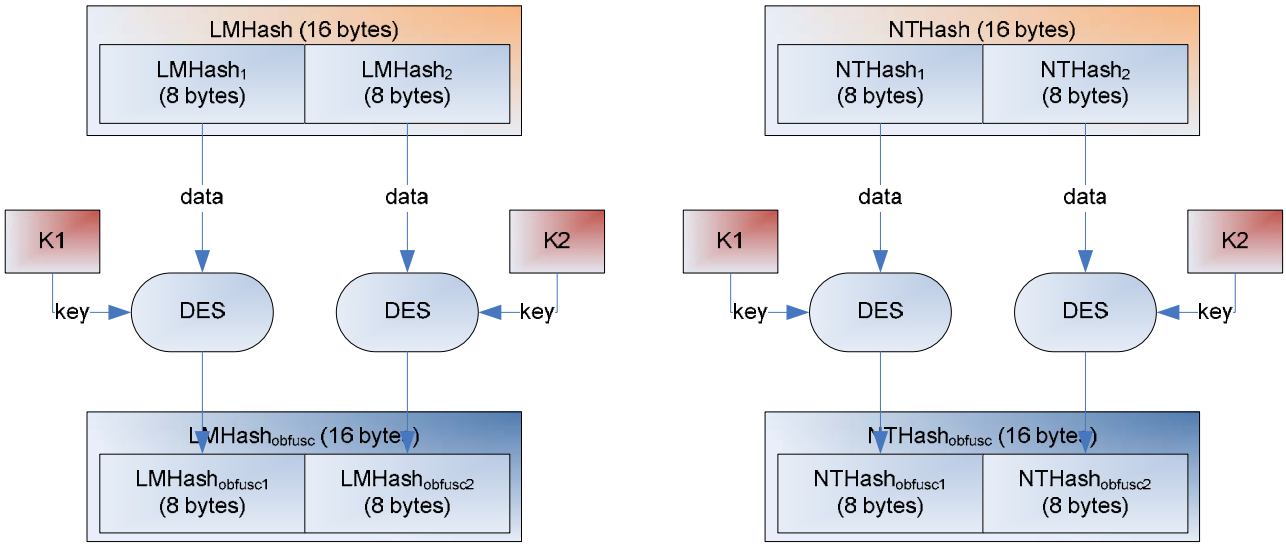


Fig. 3 – The 2nd encryption level.

The keys *K1* and *K2* used to encrypt the first and respectively the second half of the hashes resulted after applying the 1st encryption level are obtained from the user *RID* in the following manner:

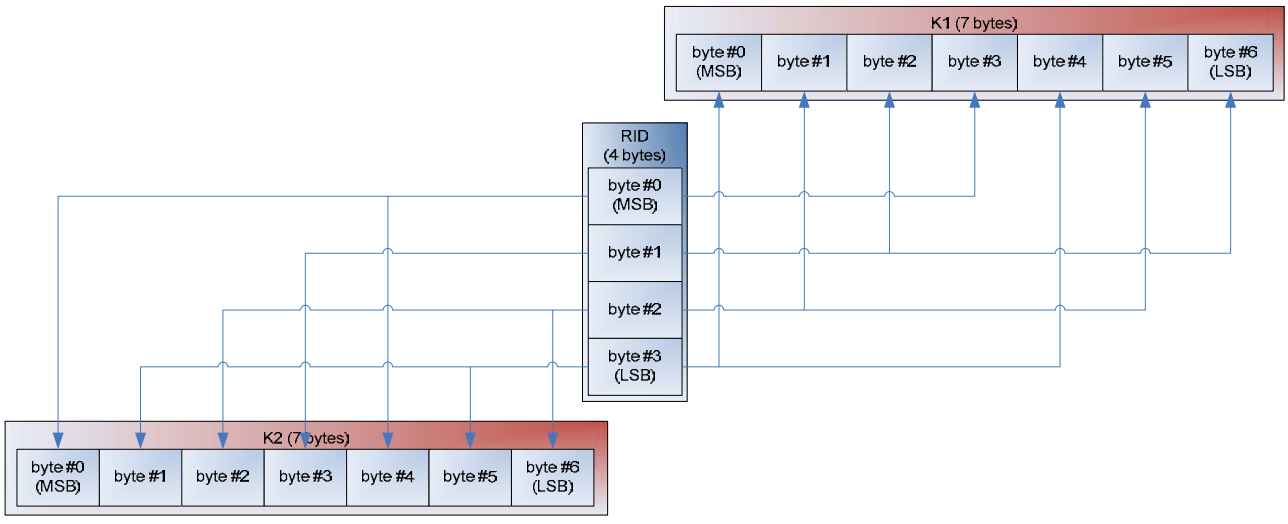


Fig. 4 – Obtaining the encryption keys for the 2nd encryption level.

By applying this level of encryption, even if two different users have the same password, the obfuscated hashes will be different because the users have different *RID* values.

The 3rd encryption level uses the most cryptographic functions, as we see in the picture below.

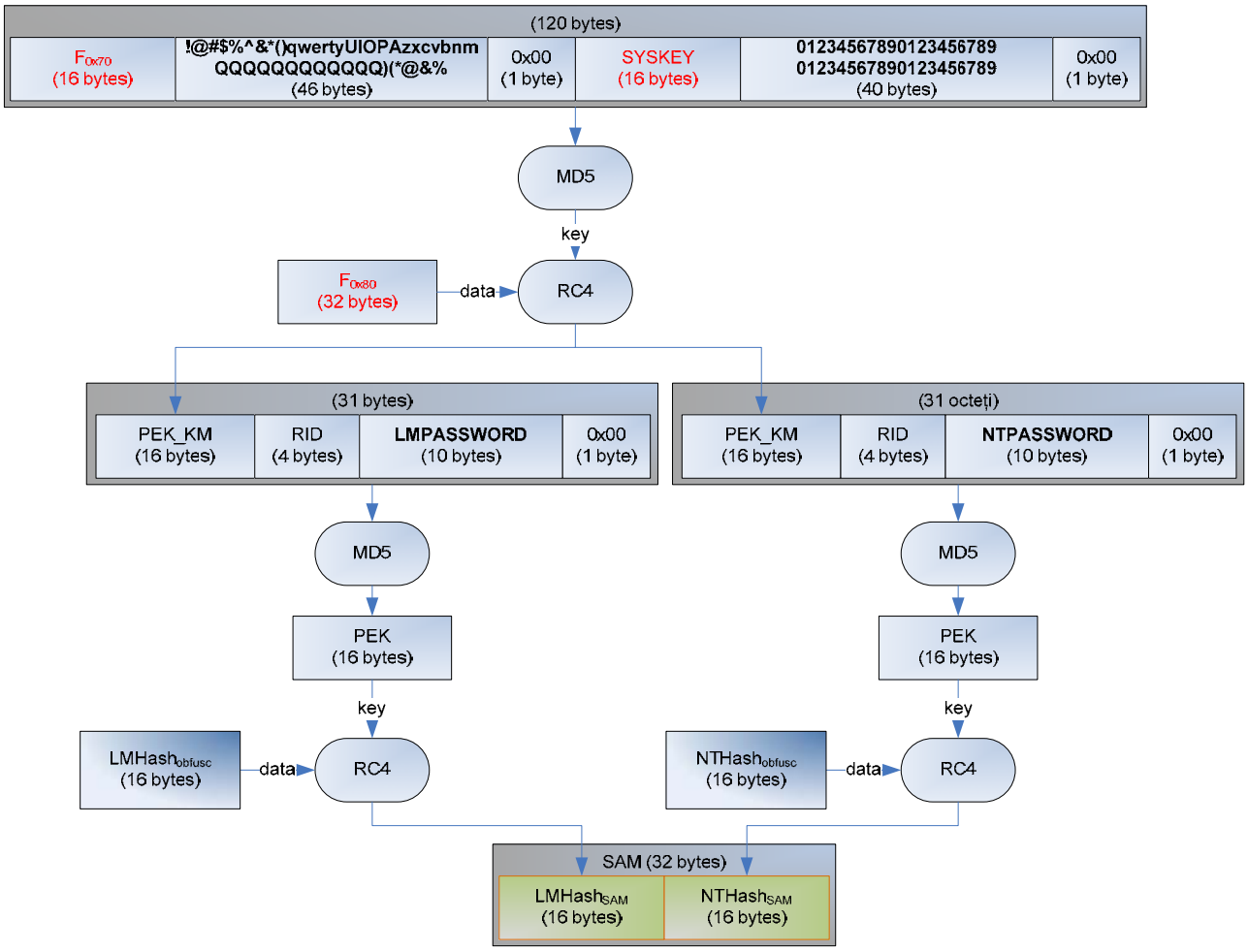


Fig. 5 – The 3rd encryption level.

Now, the only unknown values from this scheme are: *SYSKEY,* F_{0x70} and F_{0x80}.

The information we need in order to compute the *SYSKEY* value is usually stored in Windows registry, under *HKEY_LOCAL_MACHINE\System\CurrentControlSet\Control\Lsa.* The *SecureBoot REG_DWORD* specifies if *SYSKEY* is stored in registry, if it is derived from a password set by administrator or if it will be stored on a Floppy Disk.

The registry key above contains four subkeys (*JD, Skew1, GBG* and *Data*), with four bytes each. In order to obtain the *SYSKEY* value, these values are concatenated and permuted by the following rules:

P[i] = {8, 10, 3, 7, 2, 1, 9, 15, 0, 5, 13, 4, 11, 6, 12, 14}
SYSKEY[P[i]] = [JD||Skew1||GBG||Data][i]

The values F_{0x70} and F_{0x80} represent 16 and respectively 32 bytes taken from the offsets *0x70* and *0x80* from the *F* value under the subkey *HKEY_LOCAL_MACHINE\SAM\SAM\Domains\Account.*

The final values, $LMHash_{SAM}$ and $NTHash_{SAM}$ are the actual hashes stored in registry.

4. THE ATTACK

Attacking the encryption process of user passwords can be done in three steps:

Step 1: extracting the necessary data from the registry:
– *JD, Skew1, GBG* and *Data*, in order to compute *SYSKEY*;
– F_{0x70} and F_{0x80};
– $LMHash_{SAM}$ or $NTHash_{SAM}$.

In order to achieve this, we have implemented a Windows service running under the System account in order to be able to access and extract these values.

As an example, we demonstrate how to break an Administrator password, so the values $LMHash_{SAM}$ and $NTHash_{SAM}$ were read from the *V* value under the registry key *HKEY_LOCAL_MACHINE\SAM\SAM\Domains\Account\Users\000001F4*. The other values were taken from *HKEY_LOCAL_MACHINE\System\CurrentControlSet\Control\Lsa* and *HKEY_LOCAL_MACHINE\SAM\SAM\Domains\Account.*

Here are the actual keys extracted from the registry:

HKEY_LOCAL_MACHINE\System\CurrentControlSet\Control\Lsa\JD:
```
84 4E 40 71
```

HKEY_LOCAL_MACHINE\System\CurrentControlSet\Control\Lsa\Skew1:
```
42 E4 22 9D
```

HKEY_LOCAL_MACHINE\System\CurrentControlSet\Control\Lsa\GBG:
```
F4 D2 49 22
```

HKEY_LOCAL_MACHINE\System\CurrentControlSet\Control\Lsa\Data:
```
D5 A7 1F A2
```

HKEY_LOCAL_MACHINE\SAM\SAM\Domains\Account\F:
```
02 00 01 00 00 00 00 00 30 AE 6E 84 7B 68 C6 01
2E 00 00 00 00 00 00 00 00 00 00 00 40 DE FF FF
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 80
00 CC 1D CF FB FF FF FF 00 CC 1D CF FB FF FF FF
00 00 00 00 00 00 00 00 F0 03 00 00 00 00 00 00
00 00 00 00 00 00 00 00 01 00 00 00 03 00 00 00
01 00 00 00 01 00 01 00 01 00 00 00 38 00 00 00
F1 1B C8 92 CB 0B 0C 60 28 52 EC 58 80 2E F3 36
A7 51 FD DA 49 8C 0B CD D8 7D 52 76 56 A4 CA 72
78 7E 30 17 35 38 A7 73 C1 7D D9 D1 12 DB 62 D9
00 00 00 00 00 00 00 00 01 00 00 00 38 00 00 00
55 DC 56 3C 85 53 A0 73 D6 76 E4 9D FE F7 C5 20
0D 89 95 E0 7D 99 37 EA B4 2D E0 42 A4 61 31 53
6A 31 7A 38 F5 5C C9 22 AE 44 E1 23 06 21 4A 06
00 00 00 00 00 00 00 00 02 00 00 00 00 00 00 00
```

So, the values of F_{0x70} and F_{0x80} are:

F_{0x70} = `F1 1B C8 92 CB 0B 0C 60 28 52 EC 58 80 2E F3 36`

F_{0x80} = `A7 51 FD DA 49 8C 0B CD D8 7D 52 76 56 A4 CA 72`
`78 7E 30 17 35 38 A7 73 C1 7D D9 D1 12 DB 62 D9`

HKEY_LOCAL_MACHINE\SAM\SAM\Domains\Account\Users\000001F4\V
```
00 00 00 00 BC 00 00 00 02 00 01 00 BC 00 00 00
1A 00 00 00 00 00 00 00 D8 00 00 00 00 00 00 00
00 00 00 00 D8 00 00 00 6C 00 00 00 00 00 00 00
44 01 00 00 00 00 00 00 00 00 00 00 44 01 00 00
00 00 00 00 00 00 00 00 44 01 00 00 00 00 00 00
00 00 00 00 44 01 00 00 00 00 00 00 00 00 00 00
44 01 00 00 00 00 00 00 00 00 00 00 44 01 00 00
00 00 00 00 00 00 00 00 44 01 00 00 00 00 00 00
00 00 00 00 44 01 00 00 15 00 00 00 A8 00 00 00
5C 01 00 00 08 00 00 00 01 00 00 00 64 01 00 00
14 00 00 00 00 00 00 00 78 01 00 00 14 00 00 00
00 00 00 00 8C 01 00 00 04 00 00 00 00 00 00 00
90 01 00 00 04 00 00 00 00 00 00 00 01 00 14 80
9C 00 00 00 AC 00 00 00 14 00 00 00 44 00 00 00
02 00 30 00 02 00 00 00 02 C0 14 00 44 00 05 01
01 01 00 00 00 00 00 01 00 00 00 00 02 C0 14 00
FF FF 1F 00 01 01 00 00 00 00 00 05 07 00 00 00
```

```
02 00 58 00 03 00 00 00 00 00 14 00 5B 03 02 00
01 01 00 00 00 00 00 01 00 00 00 00 00 00 18 00
FF 07 0F 00 01 02 00 00 00 00 00 05 20 00 00 00
20 02 00 00 00 00 24 00 44 00 02 00 01 05 00 00
00 00 00 05 15 00 00 00 C2 3B F0 34 B0 BE 8D 6D
49 94 37 B1 F4 01 00 00 01 02 00 00 00 00 00 05
20 00 00 00 20 02 00 00 01 02 00 00 00 00 00 05
20 00 00 00 20 02 00 00 41 00 64 00 6D 00 69 00
6E 00 69 00 73 00 74 00 72 00 61 00 74 00 6F 00
72 00 00 00 42 00 75 00 69 00 6C 00 74 00 2D 00
69 00 6E 00 20 00 61 00 63 00 63 00 6F 00 75 00
6E 00 74 00 20 00 66 00 6F 00 72 00 20 00 61 00
64 00 6D 00 69 00 6E 00 69 00 73 00 74 00 65 00
72 00 69 00 6E 00 67 00 20 00 74 00 68 00 65 00
20 00 63 00 6F 00 6D 00 70 00 75 00 74 00 65 00
72 00 2F 00 64 00 6F 00 6D 00 61 00 69 00 6E 00
FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
FF FF FF FF FF BA 94 D8 01 02 00 00 07 00 00 00
02 00 01 00 B2 2E F9 E6 15 F9 CC ED 53 0E 9F 5B
B5 4F D6 56 02 00 01 00 90 F2 03 0E 6B 71 A1 8F
90 62 FA C7 50 CA E5 18 02 00 01 00 02 00 01 00
```

Final hashes' offset is computed as *V[0x9c] + 0xcc = 0x0164 + 0xcc = 0x230*. So, the two hashes, named $LMHash_{SAM}$ and $NTHash_{SAM}$, are:

$LMHash_{SAM}$ = `B2 2E F9 E6 15 F9 CC ED 53 0E 9F 5B B5 4F D6 56`

$NTHash_{SAM}$ = `90 F2 03 0E 6B 71 A1 8F 90 62 FA C7 50 CA E5 18`

Step 2: "reducing" the encryption levels. This step is about inverting the 3rd and the 2nd levels in order to obtain *LMHash* and *NTHash* (the values resulted after the 1st encryption level).

This step is necessary in order to reduce the complexity and the number of transformations that must be executed in a brute-force attack.

By looking at the cryptographic transformations in the last two levels, we can observe that:

– the 3rd level can be reduced by using the data extracted in the 1st step of the attack and the constants of the 3rd level of the encryption scheme. First, we can compute the *RC4* key, by knowing F_{0x70} and the computed *SYSKEY* value. Then, knowing F_{0x80} we can apply the *RC4* function to obtain *PEK_KM*. Next, using the known *RID* of the user (*0x000001F4* for Administrator) we apply the *MD5* hash on the concatenated values as presented in the 3rd level scheme in order to compute *PEK*, the key used to encrypt $LMHash_{obfusc}$ and $NTHash_{obfusc}$. So, by applying the *RC4* algorithm on the hashes extracted from the registry in the 1st step and using the key computed above, we finally reduce the 3rd encryption level and obtain $LMHash_{obfusc}$ and $NTHash_{obfusc}$;

– the 2nd level can also be reduced, observing that the keys *K1* and *K2* used in the *DES* encryption can be easily computed, knowing the user's *RID*. After computing these keys, we apply the inverse of the *DES* algorithm on $LMHash_{obfusc}$ and $NTHash_{obfusc}$ obtained above, obtaining the *LMHash* and *NTHash*.

After reducing the 3rd and the 2nd encryption levels on the values extracted in the 1st step of the attack, we obtain the following values of *LMHash* and *NTHash*:

LMHash = 90 94 83 75 BA 1B FE 9A AA D3 B4 35 B5 14 04 EE

NTHash = 49 E4 AA 6C 65 AD DA 82 4E DB 19 21 E5 EF E7 27

Step 3: brute-force attack on the 1st encryption level in order to break the account password.

The brute-force attack is applied on *LMHash* or *NTHash* in this way:

– attacking *LMHash* consists in finding 2 *DES* encryption keys, in 2 independent steps: the 1st key determines the first 7 characters of the password and the 2nd key determines the last N-7 characters of the password, N being the length of the password;

– attacking *NTHash* consists in finding the string for which the *MD4* hash of its Unicode conversion equals *NTHash*.

In order to process large password subdomains in parallel for reducing the computational time, one may implement this step using fast parallel technologies such as FPGA, CUDA or other distributed implementations.

We implemented the 3rd step of attacking the *NTHash* using the CUDA technology on an *nVidia® Quadro FX 3700* graphic card [6]. The picture below presents the results of brute-forcing the *NTHash* resulted above after reducing the encryption levels. The password was found in less than 13 minutes, the average processing speed being 72.84 million passwords per second. The same application can be rewritten in order to brute-force DES instead of MD4, thus attacking the *LMHash* instead of *NTHash.*

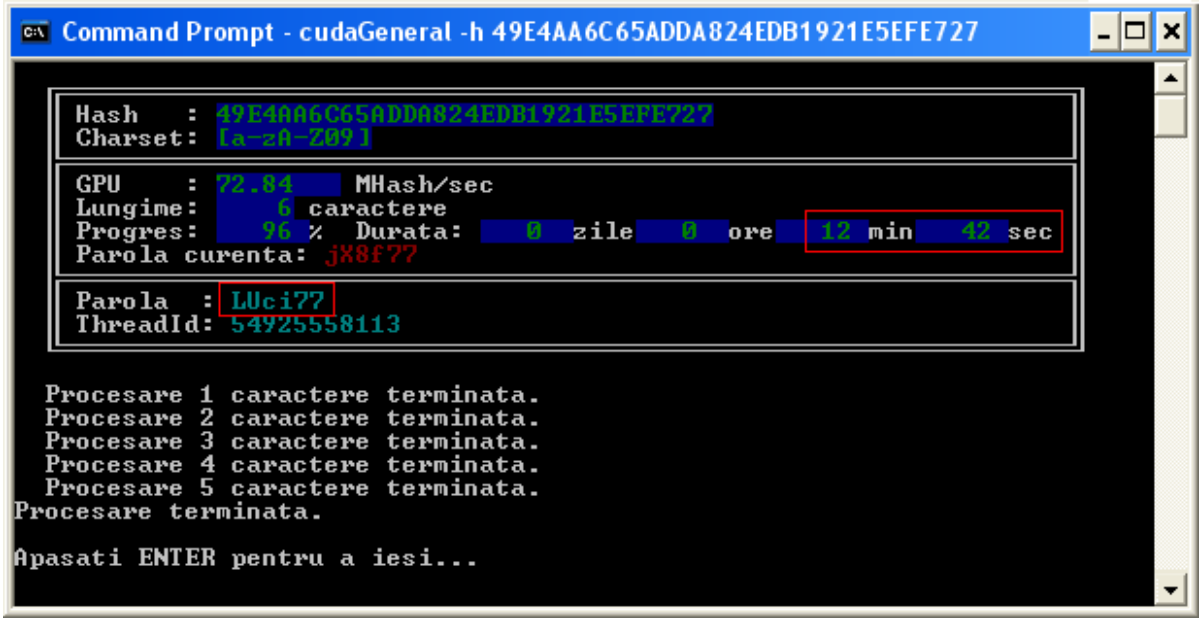


Fig. 6 – The 3rd step of the attack: brute-force example in CUDA implementation (attacking NTHash).

We have implemented a console application (*cudaGeneral*) in order to apply a brute force attack for the NTHash algorithm (that is a brute force attack against the MD4 hash of the Unicode password). The application is written in Microsoft Visual Studio, using the C++ language, a CUDA driver and library, and it is compiled using *nvcc* (NVIDIA CUDA Compiler).

The graphic card is programmed using C-CUDA, which extends the C language by allowing the definition of some C functions, called *kernels*. When called, these kernels are executed in N parallel CUDA threads, as opposite to classical functions which are executed only once [5].

A *thread ID* is automatically assigned to each thread. This ID can be accessed inside the kernel through the *threadIdx* variable (which is a 3-dimensional array). The threads are grouped into *thread blocks* which can be vectors, matrices or fields, depending on the definition of the thread.

As an example, the sum of two matrices A and B can be implemented as follows [4]:

```
// Kernel definition
__global__ void MatAdd(float A[N][N], float B[N][N], float C[N][N])
{
    int i = threadIdx.x;
    int j = threadIdx.y;
    C[i][j] = A[i][j] + B[i][j];
}

int main()
{
    // Thread block dimension
    dim3 dimBlock(N, N);

    // Calling the Kernel
    MatAdd<<<1, dimBlock>>>(A, B, C);
}
```

The number of threads in a block is limited by the memory resources of the processing cores. On current GPUs, a thread block can contain up to 512 threads.

However, a kernel can be executed on multiple blocks, so the total number of threads equals the number of blocks multiplied by the number of threads per block.

These blocks are organized into uni-directional or bi-directional block grids. The grid dimension is defined by the first parameter of the <<<...>>> syntax.

The code above transforms as follows:

```
// Kernel definition
__global__ void MatAdd(float A[N][N], float B[N][N], float C[N][N])
{
    int i = blockIdx.x * blockDim.x + threadIdx.x;
    int j = blockIdx.y * blockDim.y + threadIdx.y;
    if (i<N && j<N)
        C[i][j] = A[i][j] + B[i][j];
}

int main()
{
    // Thread block dimension
    dim3 dimBlock(16, 16);

// Grid dimension
    dim3 dimGrid( (N + dimBlock.x - 1)/dimBlock.x, (N + dimBlock.y - 1)/dimBlock.y );

    // Calling the Kernel
    MatAdd<<<dimGrid, dimBlock>>>(A, B, C);
}
```

In the CUDA technology, for the host (CPU) and the device (GPU) are allocated different memory spaces: global, shared, texture, local or registers. The developer must use carefully the read and write calls from/into these types of memory, as the performance of these actions will vary depending of the memory type and the overall performance will be affected.

The CUDA project contains three main files:

– *cudaGeneral.cu* – contains „device code", executed on GPU. This file will include the kernel and other device functions defined in the next file;

– *cudaGeneral_kernel.cu* – contains the kernel and other „device code", executed on GPU. The code declares the following device memory spaces:

```
__device__ __constant__ char charsetGPU[128];
__device__ __constant__ unsigned int charsetLenGPU;
__device__ __constant__ unsigned long targetHashGPU[4];
__device__ __constant__ unsigned long long idStartPassGPU;
__device__ __constant__ unsigned int passwordLenGPU;
```

charsetGPU[128]: constant memory, containing the working charset
charsetLenGPU: charset length
targetHashGPU[4]: the hash to be attacked
idStartPassGPU: the Id of the first password that will be processed in the current kernel group
passwordLenGPU: the length of the current passwords in process

Functions	Description
void init_GPU_constant_memory (char *charset, unsigned int charsetLen, unsigned char *hashBytes)	Initialization of the constant memory with: – charset – charset length – hash to attack
void init_GPU_constant_memory_id (unsigned long long idStartPassCPU)	Initialization of the constant memory with: – the Id of the first password in the kernel group
void init_GPU_constant_memory_passLen (unsigned int passwordLen)	Initialization of the constant memory with: – password length

(continued)

int print_device_info(bool bDisplayDeviceInfo)	Display CUDA info of the GPU
__device__ bool Transform_md4_GPU(unsigned char *src)	Executed on GPU. Compute the MD4 of the current Unicode password and compares with the hash to attack
__device__ bool Transform_md5_GPU(unsigned char *src)	Executed on GPU. Compute the MD5 of the current password and compares with the hash to attack
__global__ void Brute_Kernel_GPU_NTHash(unsigned long long * resultGPU)	The kernel that calls the hash processing of the current Unicode password and signals the CPU if a match is found
__global__ void Brute_Kernel_GPU_md5(unsigned long long * resultGPU)	The kernel that calls the MD5 processing of the current password and signals the CPU if a match is found

– *cudaGeneral_fnc.cpp* – contains C functions executed by CPU.

Function	Description
void hexAscii_to_bytes(char *src_ascii, unsigned char *dest_bytes)	Convert hex string to byte array
void tid_to_password(unsigned long long tid, char *charset, char *password, unsigned int passwordLen)	Convert the thread Id into the password to process

Assigning different passwords to the processing threads is performed as follows:

```
// thread Id
unsigned long long tid = idStartPassGPU + blockDim.x * blockIdx.x + threadIdx.x;

// aux thread Id
unsigned long long tid_aux = tid;

// pass length
unsigned int passwordLen = passwordLenGPU;

// charset length
unsigned int charsetLen = charsetLenGPU;

// loop Id
unsigned int i = 0;

//9 (pass) || 1 (0x80)
unsigned char curentPass[10] = {0, 0, 0, 0, 0, 0, 0, 0, 0, 0};

// password to be processed by the current thread
while (i < passwordLen)
{
	curentPass[i++] = charsetGPU[tid_aux % charsetLen];
	tid_aux /= charsetLen;
}

// append one bit of '1'
curentPass[i] = 0x80;

// if the computed hash equals the input one, the GPU signals the CPU
// by using the variable resultGPU from global memory (carefully use
// this variable because it drastically decreases the performance)
if (Transform_md4_GPU(curentPass))
{
	resultGPU[0] = 1;
	resultGPU[1] = tid;
}
```

Each thread creates its own password and compares the resulting hash with the target hash stored in Constant Memory. If these are equal, the device function associated with the thread will return a true value to the kernel and the kernel writes the result into the corresponding Global Memory space. In this way, the GPU signals the CPU that the password was found by the thread with the ID *tid*.

Finally, the CPU calls the conversion function (void tid_to_password(unsigned long long tid, char *charset, char *password, unsigned int passwordLen)) and displays the Built-In Administrator password found by the GPU.

5. CONCLUSIONS

Although *LMHash* has its own disadvantages over *NTHash* (reduced number of characters due to uppercase conversion, susceptible to known plaintext attacks, lacks the source of randomness, independent processing of password halves etc.), the bit level permutations in *DES* makes attacking the 1st encryption level harder and more challenging to implement than *MD4*, making the 3rd step of the attack more time consuming in the process of brute-forcing the password.

The time to finalize the attack depends with a higher degree on the 1st encryption level (the 3rd step of the attack). The other two encryption levels can be reduced in a small amount of time as long as the necessary values (F_{0x70}, F_{0x80}, *JD, Skew1, GBG* and *Data*) are extracted from the protected registry keys on the attacked machine.

The last two steps of the attack can be conducted offline, thus no interaction with the attacked computer or network is needed.

Most of the published papers related to defense techniques against password cracking recommend disabling the *LM* hashing or enabling the use of *SYSKEY*. In this paper, we presented a practical attack that works even if these recommendations are applied. One of the best security measures still remains the use of complex passwords that are frequently changed.

REFERENCES

1. Dobromir Todorov, *Mechanics of User Identification and Authentication – Fundamentals of Identity Management*, Auerbach Publications – Taylor & Francis Group, 2007.
2. Todd Sabin, *BindView Security Advisory*, http://packetstorm.foofus.com/9912-exploits/bindview.syskey.txt, 2009.
3. http://xfocus.net/articles/200306/550.html.
4. Nvidia Cuda™, *Programming Guide, Version 2.2.1*, 2009.
5. David Kirk, Wen-Mei Hwu, *CUDA (Ch. 1 to 7),* 2006-2008.
6. http://www.nvidia.com/cuda.
7. http://en.wikipedia.org/wiki/LM_hash.

Received July 25, 2013

THE PUBLISHING HOUSE
OF THE ROMANIAN ACADEMY

PROCEEDINGS OF THE ROMANIAN ACADEMY, Series A,
Volume 14, Special Issue 2013, pp. 328–337

SEMI-FRAGILE WATERMARKING BETWEEN THEORY AND PRACTICE

Mihai MITREA, Marwen HASNAOUI

Institut Mines-Telecom; Telcom SudParis, ARTEMIS Departement
9, rue Charles Fourrier, 91011 Evry France
Corresponding author: Mihai MITREA, E-mail: mihai.mitrea@telecom-sudparis.eu

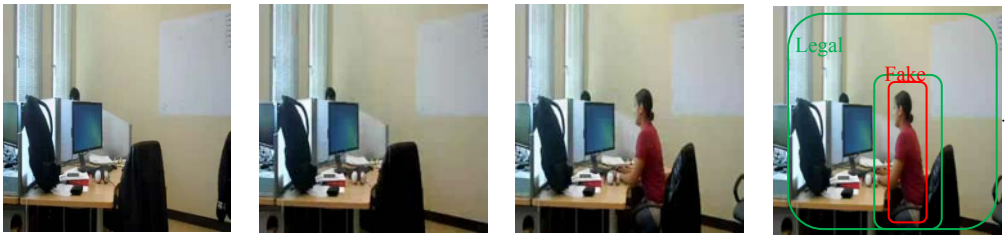
The present paper reports on a theoretical and experimental study on the possibility of achieving content-based MPEG-4 AVC video authentication. The authentication signature is extracted so as to both maximize the probability of correct decision and the mutual information under content-preserving attacks. The experiments (carried out on 1h of video surveillance corpus) demonstrate that the content-modification attacks can be identified with *Precision* and *Recall* rates larger than 0.9 while ensuring a 9s temporal accuracy and an 1/81 frame size spatial accuracy.

Key words: video surveillance, integrity, semi-fragile watermarking, MPEG-4 AVC.

1. INTRODUCTION

Nowadays, digital video is the core of various applications, ranging from home entertainment (*e.g.* video on demand) to civil/state security (*e.g.* video surveillance). Despite the particular type of application, the source/content authentication is always required. On the one hand, video on demand platforms distribute their content under strict copyright constraints: hence, the traceability of source/intermediate distribution points should be kept. On the other hand, video surveillance helps police investigations and may subsequently serve as a piece of evidence in courts: hence, both its source and content should be certified.

Two approaches can be considered in order to ensure video authenticity, namely data and content based. The former considers the data (binary) representation of the video and extracts an authentication signature (*e.g.* a hash function) which meets the uniqueness and sensitivity (fragility) requirements. This signature is further stored as metadata. The latter no longer targets the binary representation of the video but its visual/semantic content. In order to be effective, such a signature should be robust to content preserving alterations and sensitive to content changing alterations. Consider the example in Fig 1. Fig 1.a represents an original content while Fig 1.b shows its JPEG compressed version at quality factor $Q = 70$; Fig 1.c gives a content-modified version of the image in Fig. 1.b in which a person has been added. From the binary representation point of view, the three images in Figs. 1.a, 1.b and 1.c are completely different, hence their data-based signatures should be different. However, from the semantic content point of view, Fig 1.a and Fig 1.b are identical while differing from Fig 1.c. Consequently, the content-based signatures corresponding to Figs. 1.a and 1.b should be identical while differing from the signature extracted from Fig. 1.c.



(a) (b) (c) (d)

Fig. 1 – The original frame (a) suffers a content preserving attack (a compression) (b) then a content alteration attack (the insertion of a person) (c). Signature based integrity verification should discriminate between the legal/fake areas (d).

Content-based authentication methods can be classified as passive or active [1].

Passive authentication, also called forensic analysis [2] attempts to verify whether the content has been altered or not by using a direct statistical analysis. This way, no a priori processing/storing operation is required for the original content. While very appealing by its functional principle, passive authentication is not always possible and doubts about its reliability and security are raised for some applications [2].

This is not the case of the active approach, where the integrity of digital content is ensured by the embedding of an authentication signal (or signature) in the content itself, before its sharing/distribution. The active authentication is also called watermarking based authentication. Consider again the example in Fig. 1 and assume that Fig. 1.a carries (in a visual imperceptible way) a content authentication signature. On the one hand, this signature should be robust to compression; hence it should be recovered from the unaltered content areas in Fig. 1.c. On the other hand, it should be fragile against content-modifications; hence, it should no longer be recovered from the areas where the person was inserted. This way, the legal/faked areas can be discriminated into a same video frame (see Fig. 1.d).

From the applicative point of view, the main semi-fragile watermarking difficulties are the choice of the signature and of the embedding technique. The signature must ensure the semi-fragility property by being both sensitive to content changing alterations (frame cropping, object removing, …) and robust to content preserving alterations (transcoding, resizing, ...). As video sequences are preponderantly stored and distributed in compressed format, the signature should be both generated and inserted directly from/in the compressed domain, thus avoiding the computational overhead required by the decompression/compression.

In this paper, we focus on the MPEG-4 AVC (Advanced Video Coding) [3] integrity verification. First, the MPEG-4 AVC syntax elements that jointly reflect the semantic content and ensure the semi-fragility propriety are identified. The theoretical supported is granted by information theory tools. Secondly, the *m*-QIM (*m*ultiple symbols Quantization Index Modulation) insertion technique (whose optimality in proved in [4]) is considered in order to insert the mark. The experiments are carried out on 1 hour of video surveillance content and show that the proposed video watermarking based video integrity verification system is able to distinguish between content preserving and video content changing alterations.

2. RELATED WORK

Watermarking based integrity verification was already the subject of several studies [5-10]. Several insertion domains are considered: still images [5], MPEG-1/2 [6, 8, 9] and MPEG-4 AVC [7, 10].

Titman [5] and Queue [6] use image edges and corners to generate the authentication signature. The signature is embedded according to additional modification rules of overlaying 8x8 blocks. They show that these features are sensitive to content changing alteration. Beside this advantage, the method is still fragile against compression and scaling. Moreover, such deployed signatures require complex generation operations and remain of large size, thus imposing particular constraints on the watermarking insertion method.

In [7], the signature generation is based on a chaotic system. Temporal authentication information (frame index and GOP index – Group of Pictures) is used to compute the signature for each *I* (Intra) frame. The experiments carried out on a video sequence of 795 frames proved the detection of temporal changes, but the properties of spatial detection of alterations have not been evaluated. The robustness to compression (up to 30%) was also shown.

S. Thiemert *et al.* [8] calculate the authentication signature in the uncompressed domain, from the points of interest obtained through the Moravec operator [11]. A binary mask is generated for each frame *I*, then embedded into the high-frequency DCT coefficients belonging to adjacent frames. The method detects content changing alterations (object removing/inserting) while being robust to content preserving manipulation (compression up to 50%, scaling). However, the authentication signature calculation increases the method complexity. The study in [9] resumes and extends these principles. The difference relies on the use of the entropy of gray level in groups of blocks to generate the binary signature. The advanced method is robust against compression down to 50% and detects content changing alterations (object removing). Nevertheless, the signature increases the complexity of the video integrity verification system.

K. Ait Saadi *et al.* [10] consider a signature generated from low frequency quantized DCT coefficients. For each *I* frame, the low frequency DCT coefficients are collected in a buffer to be further hashed using an

MD5 function. This results into a 128 bit binary signature. The obtained signature is embedded in the *P* (Predicted) and *B* (Bidirectional) motion vectors. Experiments show that this system remains fragile to all manipulations. Moreover, the signature generation requires an MPEG-4 AVC entropic decoding.

The state-of-the art brings to light that the trade-off between fragility, robustness and complexity is not yet reached in the compressed domain. Moreover, the signature is heuristically generated, without any theoretical support.

In order to minimize the overhead induced by the decoding/encoding operations, the present study generates the signature directly from the MPEG-4 AVC syntax elements. From the theoretical point of view, the optimal syntax elements are identified by a study carried out on the basis of information theory, *cf.* Section 3. From the applicative point of view, a semi-fragile system based on this optimal signature and the *m*-QIM insertion rule is assessed within the SPY ITEA2 project, *cf.* Section 4. Conclusions are drawn and perspectives are opened in Section 5.

3. THEORETICAL INVESTIGATION ON THE AUTHENTICATION SIGNATURE

This section investigates the MPEG-4 AVC compressed stream, in order to build a syntax element based signature. The targeted signature must meet the trade-off between the robustness against content preserving and the sensitivity to content changing alterations.

3.1. Syntax elements identification

MPEG-4 AVC [3] video sequences are structured into group of pictures (GOP). A GOP is constructed by fixed number of successive images of three main types (*I*, *P*, and *B*) as illustrated in Fig 2.

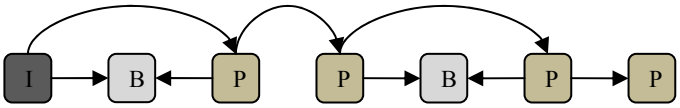


Fig. 2 – A GOP structure example.

An *I* frame describes a full image coded independently, containing only references to itself. Secondly, the unidirectional predicted frames *P* use one or more previously encoded/decoded frames as reference for picture encoding/decoding. Finally, bidirectional predicted frames *B* consider in their computation both forward and backward reference frames. According to the coding principles, *I* frames preserve more information and require more bits for encoding than the other two types.

Our study focuses on the *I* frames. The *I* frames contain the salient visual/semantic information which is also exploited by the *P* and *B* frames in that GOP. Consequently, extracting the signature from the *I* frames has two main *a priori* advantages: the signature can be related to the video semantic and represent the whole GOP. Fig 3 details the *I* frame encoding block diagram. MPEG-4 AVC transforms the uncompressed data in a classical compression chain: prediction P, transformation T, quantization Q and entropic coding E.

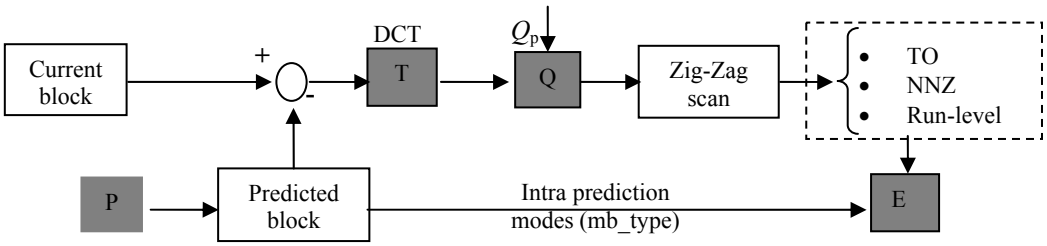


Fig. 3 – Intra frame coding diagram.

I frames are encoded according to Intra prediction modes which exploit the spatial redundancy to enhance the compression efficiency. The MPEG-4 AVC standard offers 13 directional intra prediction modes. For each current block, a predicted block is constructed from the boundary pixels of the neighboring

blocks which are previously encoded. MPEG-4 AVC offers two intra prediction modes classes: *I*16x16 (with four prediction modes) for smoothed regions and *I*4x4 (with 9 prediction modes) for the textured regions. For each block, the prediction mode minimizing the rate-distortion cost is selected.

The residual block computing the difference between the current block and the predicted block is transformed by using a DCT and a quantizer. Each quantized transformed residual block is further mapped into a 15 coefficients vector. In the baseline profile, the resulting vector is encoded by the CAVLC (Context Adaptive Variable Coding Length) encoder [3]:

– CAVLC uses the run-level entropic encoding to represent compact zero sequences.
– The non-zero values taken by coefficients are often expressed in ± 1 sequences. CAVLC encodes the ±1 number of the large compact succession of ±1 (Trailing-Ones).
– The number of non-zero coefficients (NNZ) is encoded using a lock-up table. The encoding table is determined based on the NNZ of neighboring blocks.
– Levels values of nonzero coefficients in low frequencies are larger than those in high frequencies. CAVLC takes advantage of this behavior by adapting the choice of the encoding table (VLC look-up table). The standard provides four encoding tables, depending on the NNZ value, see Tab 1.

The *I* frames contain four main syntax elements per intra block: mb_type, TO (Trailing Ones), NNZ and run-level. The extraction of these elements is performed through a syntax parser. This study will not consider the run-level. In fact, to be effective, the signature should have a limited alphabet size; this is not the case of run-level element. Tab 1 illustrates the syntax elements that will be investigated in this study.

Table 1

MPEG-4AVC syntax elements

Syntax elements	Description
mb_type	Intra prediction mode class type : (*I*4x4 or *I*16x16).
TO (Trailing Ones)	It takes a value from 0 to 3. If there is a succession of more than three ±1, only the last three are considered.
NNZ (Number of Non Zero coefficients)	NNZ encoded according to 4 tables are considered based on its value: (Table 1 (0, 1), Table 2 (2, 3), Table 3 (4, 5, 6, 7) and Table 4 (8 or more)).

3.2. Syntax elements behavior to content preserving attacks

3.2.1. Experimental protocol

This section investigates the possibility of building an authentication signature from the 3 syntax elements identified above (mb_type, TO, NNZ). To do this, the behavior of each of these elements to content preserving attacks (transcoding, Gaussian filtering, sharpening and scaling) is first investigated. Then, the considered attacks are applied according to two scenarios: (S1) the MPEG-4 AVC encoder is allowed to choose the quantization parameter *Qp* and (S2) the quantization parameter *Qp* is set to 31. Tab 2 shows the steps involved in both scenarios.

Table 2

Test scenarios

S1	S2
(1) Encode the video at variable *Qp*	(1) Encode the video at *Qp* = 31
(2) Extract the syntax elements and store them	(2) Extract the syntax elements and store them
(3) Attack the video and re-encode it according to (1)	(3) Attack the video and re-encode it according to (1)
(4) Extract the syntax elements and compare them to those extracted at (2)	(4) Extract the syntax elements and compare them to those extracted at (2)

First, the syntax elements (mb_type, TO, NNZ) are extracted using a syntax parser and stored as signatures. Secondly, the video sequence is attacked (transcoded, filtered, scaled, …) and further re-encoded according to the same initial configuration. Finally, the syntax elements are extracted from the attacked video and compared to those extracted previously from the original video. The syntax elements extracted from each 4x4 block intra are coded as follows:

$$\text{mb\_type} = \begin{cases} 0 & if\ I4\times 4 \\ 1 & if\ I16\times 16 \end{cases} \qquad \text{TO} \in \{0,1,2,3\}$$

$$\text{NNZ} = \begin{cases} 0 & if\ \ \text{NNZ} \le 1 \\ 1 & if\ \ 1 < \text{NNZ} \le 3 \\ 2 & if\ \ 1 < \text{NNZ} \le 7 \\ 3 & if\ \ \text{NNZ} > 7 \end{cases}.$$

3.2.2. Corpus

The experiments were carried out on a video surveillance corpus composed of 6 sequences of 10 minutes each, downloaded from internet [12] or recorded under the framework of the SPY project. This corpus is encoded in MPEG-4 AVC in Baseline Profile at 512 kbps, 640x480 pixel frames; the GOP size is set to 8.

3.2.3. Robustness against content preserving attacks

The initial value of a syntax element is likely to change after an attack on a random basis, given by both the attack and the content itself; hence we can investigate these modifications by modeling the attack with noise matrices. In such a matrix, the lines correspond to the values of original elements, while the columns to the values after attacks. An element in a matrix is the corresponding conditional probability, estimated on the corpus. These noise matrices are estimated by successively applying four content preserving attacks (transcoding, sharpening, Gaussian filtering and scaling) according to the two scenarios presented above. The size of the corpus was large enough so as to ensure the statistical relevance of the results: for each syntax element, 95% confidence limits are computed with relative error $\hat{\varepsilon}_r < 0.005$.

By further computing the probability of correct detection (P_c) and the mutual information (I) the related decision can be made on the optimal syntax element.

$$P_c = \sum_{i=1}^{N} P_{ii} \times P_i \,, \qquad I = \sum_{i=1}^{N}\sum_{j=1}^{N} P_{i,j} \times P_i \log(P_{i,j}/P_j) \,,$$

where $P_{i,j}$ presents the noise matrix element of coordinates i and j, P_i is the average probability that the syntax element takes the value i in the original video and P_j is the average probability that the syntax element takes the value j in the attacked video. N is the size of the corresponding alphabet (*i.e.* $N = 2$ for mb_type and $N = 4$ for NNZ and TO).

Tab 3 and 4 illustrate the distribution probability and the noise matrices for mb_type. The P_c and I values for each syntax element and for each tested attack according to the two scenarios are plotted in Figs 4 and 5, respectively. The analysis of these results brings to light two main conclusions:

– The probability of correct detection and the mutual information values show that the syntax element (mb_type) remains more robust than the two other syntax elements (TO and NNZ) against all the investigated attacks. The obtained averaged P_c over the 4 attacks and the 2 scenarios are 0.92, 0.76 and 0.47 for mb_type, NNZ and TO, respectively. I values feature an average of 0.55, 0.44 and 0.15 for mb_type, NNZ and TO, respectively;

– A better robustness is achieved for (S2). A fixed quantization step increases the mb_type correct detection probability by 0.06, 0.02, 0.04 and 0.03 and the mb_type mutual information by 0.21, 0.13, 0.01 and 0.02 for transcoding, sharpening, Gaussian filtering and scaling, respectively.

Table 3

mb_type distribution probability

	S1		***S2***	
	0	***1***	***0***	***1***
P	0.62	0.38	0.54	0.46

Table 4

mb_type transition matrix

	Transcoding				Sharpening				Gaussian filtering				Scaling			
	S1		*S2*		*S1*		*S2*		*S1*		*S2*		*S1*		*S2*	
	0	***1***	***0***	***1***	***0***	***1***	***0***	***1***	***0***	***1***	***0***	***1***	***0***	***1***	***0***	***1***
0	0.95	0.04	0.98	0.02	0.78	0.22	0.82	0.18	0.94	0.06	0.95	0.05	0.92	0.08	0.93	0.07
1	0.11	0.88	0.03	0.96	0.06	0.94	0.03	0.96	0.15	0.85	0.05	0.94	0.12	0.88	0.07	0.93

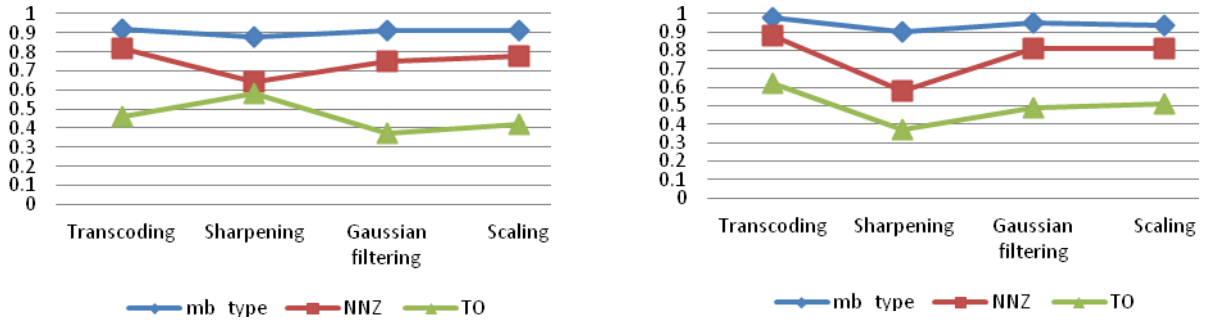


Fig. 4 – Probability of correct detection, for (S1) – left and (S2) - right.

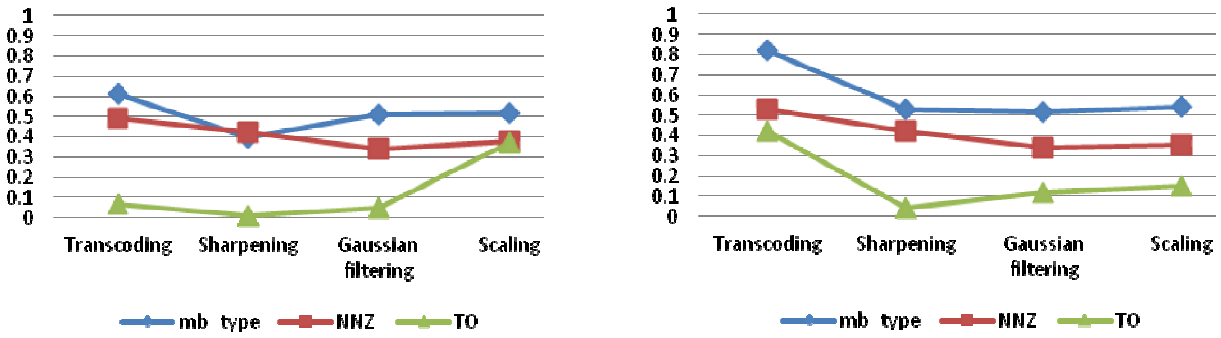


Fig. 5 – Mutual information, for (S1) – left and (S2) - right.

These results prove that the mb_type is the single element able to satisfy the constraint of robustness over the three investigated syntax elements. Thus, the rest of the study will concern only the mb_type elements.

3.3. Sensitivity to content changing alterations

This section investigates the sensitivity of mb_type syntax element to content changing alterations. In this study, the content changing alteration attack is performed by removing a person as illustrated in Fig 6-a and Fig 6-b. Just for illustration, such an attack can be performed by any non-professional user on a home PC, with software available on Internet; it takes about 10 minutes to process 1 second of video.

First *I* frame of original video (a) First *I* frame of attacked video (b) Content alterations detection (c)

Fig. 6 – Content changing alterations.

In order to spatially localize the content manipulations, an alteration detection image is generated. When the mb_type is detected as changed, the pixel value corresponding to it is set to red. The resulting detection image is illustrated in Fig 7. To spatially locate alterations, positions reported as manipulated by the detection matrix are further projected onto the inspected image, see Fig 6-c. It can be noticed the presence of false alarms and that the red block are denser in the altered area than other areas. To enhance the alteration detection and avoid the false alarms, the following filter is applied to the detection image:

$$M(i,j)=\begin{cases}1 \ \ if \ \ \sum_{i_0}^{a}\sum_{j_0}^{b} M(i+i_0, j+j_0) > s \\ 0 \qquad\qquad\quad otherwise\end{cases},$$

where M is the alteration detection image, $a \times b$ presents neighborhood window filter size and s is the decision threshold. i_0 and j_0 present the line/column indexes in the filter window. Fig 7 illustrates the evolution of the detection image as function of the decision threshold and Fig 8 reports the obtained false alarm probability as a function of $s = 3$. We note that from a given threshold ($s = 3$ in our case) false alarms disappear and the altered area is surrounded. We also notice that the optimization of the parameter s may be mandatory to improve the spatial accuracy of the alteration detection according to the targeted application.

M for *s* = 0 (a) *M* for *s* = 1 (b) *M* for *s* = 2 (c) *M* for *s* = 3 (d)

Fig. 7 – Alteration detection matrix.

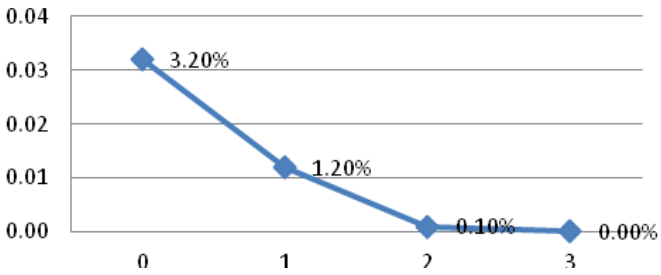


Fig. 8 – False alarm as function of s.

3.4. Conclusion

This section investigates the MPEG-4 AVC syntax elements with respect to their potential usage as authentication signature robust to content preserving attacks and sensitive to content changing. After an *a prior* study, 3 syntax elements (mb_type, NNZ and TO) are identified and tested by using information theory based entities. First, noise matrices, P_c and I demonstrate that mb_type is the optimal choice for content preserving attacks. Secondly, mb_type proved to be able to result into probability of false alarms equal to zero under content changing attacks. The next section will present a semi-fragile watermarking method whose inserted signature is based on mb_type syntax element.

4. APPLICATION VALIDATION OF THE SEMI FRAGILE WATERMARKING

This section details the deployment of mb_type based signature for building an MPEG-4 AVC video integrity verification system and subsequently evaluates its performances under the SPY[1] project framework.

4.1. Method presentation

The authentication signature is generated by encoding mb_type to an *m*-ary alphabet.

The obtained signature is further inserted by an *m*-QIM watermarking technique [4] in the video content. The low complexity requirement can be met when such a signature is extracted and inserted directly from/in the MPEG-4 AVC syntax elements, with minimal decoding/re-encoding operations. To meet this requirement, we consider individual groups of *i* successive *I* frames (further referred to as *I*-Group) sampled from an MPEG-4 AVC video sequence. The signature is computed from the first I_0 frame in such an *I*-Group. The obtained signature conveying information about the content of the first frame is further inserted into the rest $i-1$ *I* frames of that *I*-Group by the *m*-QIM embedding method. The shorter the *I*-Group, the more accurate the temporal localization of altered content.

In order to verify the integrity of an attacked *I*-Group, the authentication signature of the attacked video is computed from its first I_0 frame. This signature is compared to the mark extracted from the rest of the *I*-Group frames. In our experiments, we consider that an area in a frame is altered when at least s (*cf.* Section 3.3) elements of the attacked signature elements belonging to that area do not match with the corresponding extracted watermark elements.

4.2. Functional evaluation

4.2.1. Robustness

In videosurveillance context, transcoding is the main harmless authorized attack. While the Section 3.2.3 investigated the effects of this attack at the signature feature level, we are now assessing the global effectiveness of the video integrity system. In this respect, the watermarked sequence was subject to a transcoding attack applied according to the S2 scenario described above. In order to identify the spatial content alterations, the detection procedure was applied on areas obtained by partitioning the I_0 frames with a 9x9 equidistant rectangular grid (see Fig 9).

This set-up allows the robustness to be objectively assessed by the probabilities of missed detection (*i.e.* the probability of not detecting a watermark from an initially marked area), and false alarm (*i.e.* the probability of detecting a mark in an initially un-watermarked area).

Our experiments showed that the re-encoding from 512 kbps down to 128 kbps resulted in no content modification, thus demonstrating the robustness of the proposed system, with ideal values for the probabilities of missed detection and of false alarm (P_m , P_f).

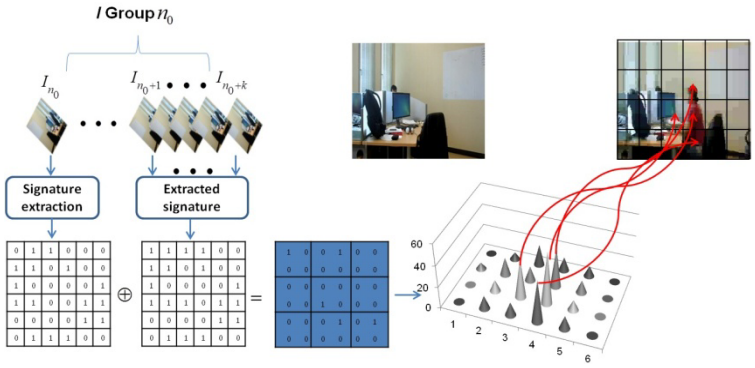


Fig. 9 – Spatial alteration detection.

[1] The ITEA 2 SPY (Surveillance imProved sYstem) European project aims at creating a new automated intelligent surveillance and rescue framework adapted for mobile environments.

4.2.2. Fragility

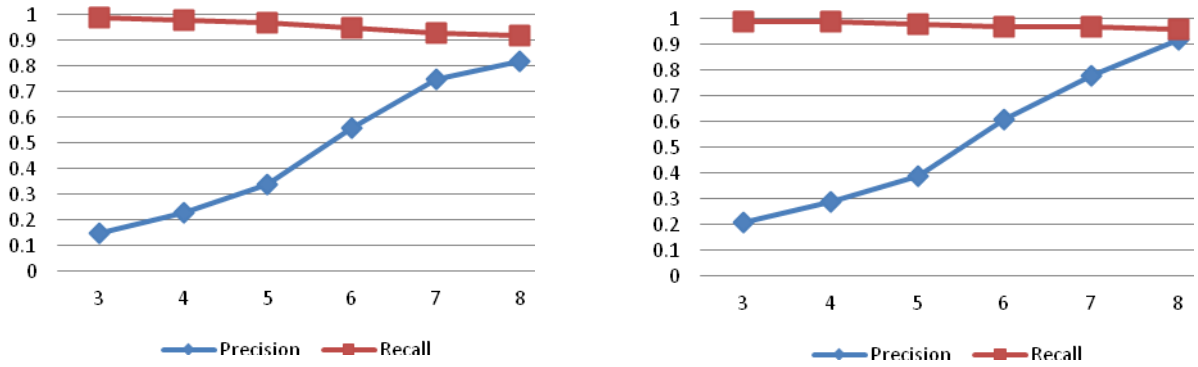
This Section investigates the sensitivity to content changing attack. The content is considered as being attacked when one object is moved, deleted or substituted. To simulate this attack, we used a piece of code that tampers the videos by arbitrarily changing 1/16 of the frame content. For each video sequence in the corpus, we applied such an attack to sequences of successive frames (between 9s and up to 3min).

In order to spatially locate alterations, we kept the same conditions as in Section 4.2.1: the I_0 frames in each *I*-Group are portioned in 81 areas, according to 9x9 equidistant rectangular grid. From the fragility point of view, an ideal watermarking method will fail in detecting the mark from each and every area which was subject to content alterations. While such a behavior can be also expressed in terms of probability of missed detection and false alarm, the literature bring to light two more detailed measures, namely the precision and the recall ratios, defined as follows [13]:

$$Precision = \frac{tp}{tp + fp}, \quad Recall = \frac{tp}{tp + fn},$$

where tp is the number of true positive (*i.e.* the number of content modified areas which do not allow the mark to be recovered), fp is the false positive (*i.e.* the number of content preserved areas which do not allowed the mark to be recovered) and fn is the false negative number (*i.e.* the number of content modified areas which allowed the mark to be detected).

Fig 10-a illustrates the obtained precision and recall average values as a function of the threshold s (*cf.* Section 3.3). The experiments exhibit $Precision = 0.81$ and $Recall = 0.92$ at $s = 8$ (*i.e.* more than 50% of area syntax element size). As these average measures are quite far from the ideal cases ($Precision = Recall = 1$), we went further in our investigation. Fig 11 illustrates the temporal detection of alterations: the abscissa corresponds to the *I-Group* index while the ordinate is set to 1 for the *I-Groups* identified by the system as being modified. The content attacked sub-sequences are circled in blue.



(a) (b)

Fig. 10 – Precision and recall as function of *s*: initial method (a) and method with post processing (b).

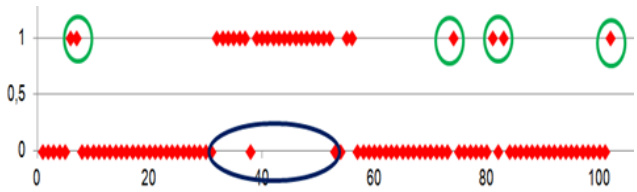


Fig. 11 – Temporal alteration detection.

Fig 11 shows that almost all altered *I*-Groups have been detected; however, some content-preserved *I*-Groups (circled in green) were also detected. When inspecting the entire corpus on the temporal axis, it was noticed that such errors are sparse. From the result interpretation point of view, feed-back provided by professional under the framework of the SPY project brought to light that when an *I*-Group is maliciously

altered, it is very likely to have other successive blocks altered. Consequently we included in our decision procedure a post-processing rule of the type: one *I*-Group is considered as altered if at least two *I*-Groups that succeed it or precede it are detected as altered. This way, the new values for precision and recall ratios are $Precision1 = 0.97$ and $Recall1 = 0.97$ at $s = 8$ (see Fig 10-b). Of course, this increase in the statistical performances was obtained at the expense of decreasing the time accuracy: content modifications shorter than 9s cannot be detected.

4.3. Conclusion

This section shows that coupling mb_type based signature and *m*-QIM insertion can ensure practical relevant results for semi-fragile watermarking based integrity verification. Experiments conducted under SPY project corpus showed that the advanced method is robust against content preserving attacks and able to locate content changing modification with a *Precision* value of 0.92 and *Recall* value of 0.97. The spatial and temporal accuracies are evaluated at 1/81 frame size and 9s, respectively.

5. CONCLUSION

With the present study, the MPEG-4 AVC syntax elements which can optimally serve the needs of content-based video authentication are first identified by carrying out an information-theory based investigation. Secondly, a semi-fragile video watermarking software is implemented and assessed in terms of robustness (against content preserving attacks) and precision in identifying the content changing attacks.

Future work is scheduled on extending this study for ensuring the authentication of the emerging HEVC (High Efficiency Video Coding) standard.

REFERENCES

1. S. Upadhyay and K. Singh, *Video authetication – an overview,* International Journal of Computer Science & Engineering Survey (IJCSES), vol. **2**, No. 4, pp. 75–96, November 2011.
2. Z. J. Geradts and J. Bijhold, *Forensic video investigation with real time digitized uncompressed video image sequences*, Proc. SPIE, vol. **3576**, p. 154–164, 1999.
3. I. Richardson, *H.264 and MPEG-4 video compression*, The Robert Gordon University, Aberdeen, 2003.
4. M. Hasnaoui, M. belhaj, M. Mitrea and F. Préteux, *MPEG-4 AVC stream watermarking by m-QIM technique*, Proc SPIE. **7881**, pp. OL1-OL8, 2011.
5. J. Titman, A. Steinmetz and R. Steinmetz, *Content based digital signature for motion pictures authentication and content fragile watermarking*, Multimedia Computing and Systems, IEEE International Conference, vol. **2**, pp. 209–213, 1999.
6. P. M. Queue, *Toward robust content based techniques for image authentication*, Multimedia Signal Processing, IEEE Second Workshop, pp. 297–302, 1998.
7. S. Chen and H. Leung, *Chaotic watermarking for video authentication in surveillance applications*, IEEE Trans On Circuits and Systems For Video Technology, vol. **18**, pp. 704–709, 2008.
8. S. Thiemert, H. Sahbi and M. Steinebach, *Applying interest operators in semi-fragile video watermarking,* Proc. SPIE, vol. **5681**, pp. 353–362, 2005.
9. S. Thiemert, H. Sahbi and M. Steinebach, *Using entropy for image and video authentication watermarks*, Proc. SPIE, vol. **6072**, pp. 18–1 – 18–10, 2006.
10. K. AIT. Saadi, A. Bouridane, and A. Guessoum, *Combined fragile wateramrking and digital signature for H.264/ AVC video authentication*, 5th Symposium on Comminucation and Mobile Network, pp. 1–4, 2010.
11. H. P. Moravec, *Towards automatic visual obstacle avoidance*, Morgan Kaufmann Publishers, 5th Int. Joint Conference on Artificial Intelligence, Vol. **2**, 1977.
12. www.webcamendirect.net
13. M. Buckland, F. Gey, *The relationship between recall and precision*, Jornal of the American Society for Information Science, vol. **45**, pp. 12–19, 1994.

Received July 1, 2013

THE PUBLISHING HOUSE
OF THE ROMANIAN ACADEMY

PROCEEDINGS OF THE ROMANIAN ACADEMY, Series A,
Volume 14, Special Issue 2013, pp. 338–356

TOWARDS SECURE NETWORK COMMUNICATIONS WITH CLIENTS HAVING CRYPTOGRAPHICALLY ATTESTABLE INTEGRITY

Dan Horea LUTAS[*], Sandor LUKACS[*], Raul Vasile TOSA[*], Andrei Vlad LUTAS[*]

[*] BITDEFENDER SRL
1 Cuza Vodă Street, Cluj-Napoca, ROMANIA
Corresponding author: Sándor LUKÁCS, E-mail: slukacs@bitdefender.com

In a client-server communication, the server side must trust the client before releases confidential data or accepts commands received from the client. While industry best practices and considerable amount of research focus on secure credential authentication, we stress that this is at most a deceptive effort, providing only a false sense of security. We underline the sharp contradiction between state-of-the-art industry wide practices and the security that researchers campaign for, with special focus on online banking solutions. We present a brief review of a wide range of cyberattack techniques used today to perform large scale identity theft, financial fraud or espionage. We believe that current approaches, including inside-OS security solutions or relying only on credential authentication are outdated, and an industry wide shift is needed to provide trustable, integrity attested clients. Our solution is created around a type 1 bare-metal hypervisor, relying on hardware-enforced technologies to provide strong isolation between a secure operating environment on the clients and a possibly compromised OS. Blending an easily deployable solution with remote cryptographic identity attestation support, we believe that our proposal carries a significant value, both from security point of view and market applicability. In order to support a proper evaluation of the strength and weaknesses of the solution, we also present a comprehensive review of various remaining attack techniques and strategies.

Key words: trusted communication, hypervisor, VMM, secure client, trusted client, remote attestation, attestable integrity, Intel TXT, malware attack techniques, vulnerabilities.

1. INTRODUCTION

We live in a world where we, our privacy and ultimately the whole global economy are becoming more and more dependent on the Internet [109]. Stock trading and bank transactions are performed online, we pay our taxes online and do a great share of our purchases on the Internet, using credit cards. There are millions of people working remotely [58], accessing confidential business information, governmental or even military data from remote clients. Industrial and military espionage has grown to unprecedented levels [16, 114].

To underline the practicality and impact of our proposal, we shall mention a few issues implying client side malware infections or cyberattacks, comprising two key domains: financial and identity fraud, and economic and military espionage. Financial institutions and citizens using online banking are among the most frequent targets of advanced malware and cyberattacks [83, 45], with a recent United Nations report saying that more than 30% of total global cybercrime is computer related fraud and forgery [118]. There are numerous examples of online banking, identity theft and fraud related, money stealing malware, in many cases with tens of millions of US dollars stolen using a single advanced malware, operated by a few persons [130, 51, 81, 26, 110]. A 2012 study done by the U.K. National Fraud Authority [136] reveals that 9.4% of all adult U.K. citizens had been an identity fraud victim in the last year, only 44.7% of the victims being able to recover their losses. On average these victims lost £481 each. Many banking Trojans have been active in the wilds for 5-8 or more years, producing enormous damage [102, 124]. On the military side, the quite recent Stuxnet, Duqu [56, 8] and Flame [137] APTs were using various client side infection methods to be able to penetrate complex networks in stages. Another recent malware, Gauss [54], was characterized as "*a nation state sponsored banking Trojan which carries a warhead of unknown designation*".

Globally, we are dealing with two major class of cyber-enemies. First of all, there are cybercriminals focusing on stealing our identity, our credentials and ultimately our money. They are behind more than half of all illicit cyberactivities [83], with global cybercrime being well over $100 billion today [107, 41]. Secondly, we are dealing with advanced persistent threats – well organized criminal groups, foreign governments or military having both the intent and the capacity to continuously attack our networks, our systems or even our critical infrastructure.

Today numerous activities and industries are relying more than ever on client-server communications, where, in a broad sense, an attacker could target three key elements: the integrity of the server (*e.g.* directly penetrating a financial institution or a military server), the integrity of the communication channel (*e.g.* intercepting network packets in man-in-the-middle attacks) or the integrity of the client (*e.g.* setup keyloggers or backdoors onto an endpoint device). Analyzing this from a practical point of view, it is important to note, that one can have invulnerable servers, unbreakable cryptography for the communication channel (*e.g.* eavesdropping and brute force attack proof), but once the client side is compromised, so it is our privacy and the security of our critical data.

It is important to note that client systems are in many cases entrance vectors for malware or advanced attacks ultimately targeting critical server systems, as it was the case for Stuxnet family for ex. This is especially true for military grade / espionage, where the ultimate target is almost priceless: there is a long and impressive history of stealing confidential data from critical military projects [16, 88, 114, 79].

In this paper we are focusing on the security and the trustworthiness of the client side. There is one fundamental question the server should ask for in any over-network client-server communication: how does the server side decide if it can trust the client or not? When can confidential information be released to the client over the network?

1.1. Authentication *vs.* integrity attestation

The straightforward answer that comes into one's mind to the previous questions is like: "*the server will ask the client to send valid credentials, comprising user name, password or some kind of token value, and no critical information will be provided, unless the credentials are valid*". Obviously, this scenario assumes that credentials have not been stolen and the communication channel has not been compromised. An immense research has been carried out on the topic. There are countless papers and technologies focused on how to properly validate credentials, how to prevent credentials being stolen, *e.g.* by keyloggers or screen capturing malware. There are whole industries focused on the securing authentication, employing numerous methods, like online banking using hardware token or SMS based two-factor authentication, or employing the state-of-the-art Intel IPT [46, 144, 49] hardware based technologies, but still failing to provide adequate protection against advanced attackers. We will give several reasons for this in Section 2.1.

There is a fundamental and clear difference between *authentication of credentials* and *attestation of integrity*. This is one of the key ideas behind our proposal. In essence, *securing authentication* ensures that no one can steal our credentials or use them to legitimate as being us, *e.g.* to pretend to be us while logging in at an online banking account. On the other hand, *remote integrity attestation* can assure the server side that the client software has not been tampered with, *e.g.* an online banking application is exactly in the same state as the bank delivered it to the client at deployment time. Although the difference is significant between the two cases, to be able to consider the client trustworthy, the client software needs to satisfy another key criteria: it must be in complete control of the client hardware.

1.2. Strengthening the client. Raising the cost of an attack

Before we can talk about highly secure client-server communications, we shall reflect on two more essential aspects. First of all, considering a system where critical data is shared across network by two or more entities, the security of data at maximum equals the security of the weakest link from the whole chain. In most organizations, client systems represent the weakest point.

Secondly, in our vision, there is no perfect security. The strength of the security provided by a given solution is directly proportional to the time and resources needed to be invested in order to break into the protected system. We could quantify the value of a security system by how much does its deployment raise

the cost of the attack needed to successfully break the system. We estimate that our proposal has the value and strength to raise the cost of the attacks by at least a factor of magnitude.

We propose a solution designed around a client oriented, hardware virtualization based type 1 bare-metal hypervisor. As Vogl [116] and Zaharia [131] observe, hypervisors offer numerous advantages, including strong insolation, a significantly reduced codebase – providing a small attack surface and the possibility to analyze or enforce security from outside the operating system. We aim to use the hypervisor to take complete control over hardware, including KVM devices (keyboard, video, mouse) and to create a secure working environment inside a custom Linux trusted virtual machine. The hypervisor will employ all known Intel VT-x/EPT, VT-d and TXT technologies, in order to properly enforce security.

2. ATTACKING THE CLIENTS, STEALING THE DATA

2.1. Advanced malware and attack techniques

One essential property of our day malware is that it tries to remain stealthy and active as long as possible inside a system. The most frequently employed technique to achieve this is the usage of kernel rootkits [105, 69]. Once installed on the target system, widespread kernel-mode rootkits can easily be used to bypass traditional security solutions. According to some reports [100], from a few known rootkit samples 10 year ago, the number of unique rootkits today is likely over the 2.5 million mark. Advanced kernel malware routinely infects millions of computers around the world. In one example, in about a year, one kernel rootkit infected over 16 million PCs [87]. The next version of that malware successfully penetrated at least 46 of the Fortune 500 companies [123].

Another particularly dangerous advanced attacks are the so called DMA (Direct Memory Access) attacks, in which, using DMA controllers from various peripheral devices (*e.g.* network or audio cards), a malware can bypass traditional CPU privilege levels and overwrite any piece of a security software [104, 91, 2]. DMA can also be used to steal critical data (*e.g.* passwords or encryption keys) from different applications, bypassing traditional protection measures. More recently DMA based attacks are integrated directly into exploitation frameworks [11], making them easily deployable in wide scale attacks.

A great number of cyberattacks today relies on poorly written software that contains vulnerabilities, which are routinely exploited to execute malicious code [44, 14]. Landwehr hits the nail on the head [62] saying that "*today, most successful attacks simply exploit flaws that were inadvertently placed in the system during development and were not removed by industrial quality control mechanisms prior to distribution*". Although there are numerous attempts to protect software against those issues (NX bit / DEP, SMEP [50], ASLR [149], sandboxing, hook based protections being the most significant of them), until today none of them resisted for a long time against malware attacks [125, 57]. Even highly elaborated and deeply built-into-OS approaches, like Microsoft's PatchGuard in 64 bit editions of Windows, have been proved to be short of expectations in the fight with advanced malware [99, 36]. Although there are various reasons why those technologies failed to deliver adequate protection, we can observe one common and very significant weakness behind all of them: they are all running at the same privilege level as the malicious code they try to protect from (*e.g.* rootkit). Although today there are more than 50,000 publicly disclosed [14], well-known vulnerabilities, the most alarming fact is that 40-50% of the new ones disclosed each year remains without proper patching [44], such that their exploitation, even at kernel level [125, 5, 13], is both easily at hand and very lucrative for cyber-criminals. We must point out, that in case of advanced attacks, based on kernel exploits or kernel rootkits, there is no way to properly protect any kind of client side application from inside the OS; that is, client side network communication can be easily eavesdropped, encryption keys or credentials stored in memory can be easily stolen.

One particularly important class of malware are Banking Trojans [18], which are behind the so called Man-in-the-Browser (MiB) attacks [80, 70]. They are widely used to infect web browsers, intercepting network communication between the user (client) and bank (server), being able to fundamentally change every aspect of online banking transactions (*e.g.* changing the destination for a transfer or changing also the amount). They usually seek additional user credentials that attackers use to perform fraudulent fund transfers. It is important to note, that MiB attacks essentially mean the corruption of integrity of the client

side application, and thus, one way to protect against them is to be able to attest the integrity of the online banking browser. A critically important aspect of MiB attacks is that they regularly bypass two-way authentication, as all fraudulent activities are carried out after the user legitimately authenticated itself. In other words, credentials do not need to be stolen at all in order for a MiB attack to steal your money. As a prime example, the alarming advance of the ZEUS Trojan triggered the European Network and Information Security Agency (ENISA) to recommend [25] the banks nothing less than to consider all client PCs are being possibly infected. Already in 2011 it was estimated, that ZEUS infected 3.6 million PCs in the U.S. alone, *i.e.* over 1% of all systems [108]. Last year, researchers found a Banking Trojan that is capable to operate completely autonomously, being able to clean the banking account of a victim without requiring user interaction at all [120]. We stress that MiB attacks are not specific only to financial institutions or online banking / shopping. For example, the same techniques can be used for long term industrial espionage also.

The use of digitally signed malware is widespread [129], underlining the weakness of the certification mechanisms. We have examples when government certificates are stolen [33], with advanced attacks using validly signed malware [8, 66] or banking Trojans easily installing on victim systems because they have valid digital signatures [141]. We have today around a million of digitally signed malware, which can silently install kernel rootkits or other advanced malicious codes into the user's machine. This is a solid argument underlying the need to start using integrity attested clients.

The use of password and identity stealing malware is very widespread [103, 27, 72]. They usually employ keyboard, mouse and screen capture software. According to one significant report [68], 100% of data breaches investigated involved some kind of credential theft.

Two-way authentication, based on digital signatures, SMS or hardware tokens is widely used today, especially by the financial industry. However, they are hopelessly broken [92, 59] in case we are dealing with sufficiently skilled attackers. There are numerous reported cases in which criminals successfully intercept both authentication channels (*e.g.* €36 million was stolen recently using mobile malware that intercepted SMS messages from two-way authentication [52]). We must point out, that sophisticated MiB attacks don't even need to bypass two-way authentication or steal the user's credentials to be able to perform fraudulent activities under the client's identity. They can hijack legitimate user sessions then alter the traffic after the user identified itself. More than this, SMS-based two-factor authentication is broken since many years, not only by mobile malware, but by direct GSM/3G traffic eavesdropping also, using relatively cheap and simple equipment available today [28, 82].

We stress that this list is not exhaustive, but only exemplificative. However, at least two more aspects need to be considered here, in order to get a more complete view of the issues we are facing. First of all, advanced cybercriminals (or people behind APT attacks) are widely using techniques to avoid anti-malware detection [34, 17, 97]. Secondly, they use heavy automation and huge computing resources in areas like detection-avoiding malware generation / polymorphization [142], password cracking [85], exploit generation [3] or vulnerability scanning. The results can be devastating, as a recent report [70] described state-of-the-art malware automation used to siphon at least $78 million from bank accounts around the world. According to one research [68], on average, a typical computer network, protected by up-to-date security solutions have been penetrated by advanced attackers 243 days before the security breach is first detected. That is, advanced attackers have an enormous time window to perform their malicious business.

2.2. Significant examples from current solutions and approaches

To better underline the contrast between our proposal and existing solutions, we will present details on two important topics, related to the integrity of clients: online banking and exploit protection solutions.

Current state-of-the-art online banking solutions and approaches, from traditional security software vendors, include the usage of sandboxed evaluation [80] and/or security hardened browsers, with examples like Kaspersky SafeMoney [55] or Bitdefender SafePay [9]. While they do protect against some of the attacks used today, they have no chance against advanced kernel malware or targeted attacks. Many banks explicitly recommend users to have up to date anti-malware and anti-phishing solutions, which again, do indeed protect against some common attacks, as we pointed out in Section 2.1 can be bypassed by advanced attacks. There are several online banking software which include virtual keyboard and screen capture prevention methods (*e.g.* using a kernel driver, thus being completely vulnerable to an advanced kernel

attack running at the same privilege level). Both, the widely used hardware token or SMS based two-factor authentication, considered today an industry best practice and innovative technologies, like Intel IPT also [106, 46], are addressing only the first part of the problem: user authentication. They however do provide effectively no protection at all against advanced banking Trojans, man-in-the-browser attacks (which steal billions of dollars each year), not even mentioning advanced kernel exploit or DMA attacks. To sum it up, current state-of-the-art approaches represent less than half of a solution. We recommend the reader to confront this with best practice recommendations, flyers presenting online banking solutions and FAQ support lists from local banks. Our estimate is, that the usage of 128/256 bit SSL, two-factor authentication based on hardware token or SMS and anti-phishing recommendations will represent the bulk of the list. We still have to see a bank talking about ZEUS grade banking Trojans, offering truly protective solutions.

For protection against exploits a whole set of user-mode and kernel-mode technologies and products have been developed. Until today none of them stands against the attacks, underlying once again the need for a radically new approach: we recommend switching from software-only to hardware-enforced isolation and security solutions. XD (eXecution Disable) / NX bit and related Microsoft DEP technologies are widely used in an attempt to prohibit instruction-fetch operations from protected pages, to prevent code execution. This is however routinely bypasses by various attacks, like return oriented exploits executing VirtualProtect API to disable protection for a given area. ASLR (Address Space Layout Randomization) techniques ensures that after each reboot the various modules will be loaded to a different base-address, thus making exploitation harder (classical exploits tend to rely on hard-coded values – pointers to APIs, to TIB – Thread Information Block, PEB – Process Environment Block etc.). ASLR can be bypassed for example by creating an algorithm that explicitly searches for the needed data structures inside memory (*e.g.* by pattern matching) instead of relying on hard-coded values [50, 139]. Various products (*e.g.* Microsoft Office, Adobe Reader or Google Chrome) employ web or document specific sandboxes and low credentialed processes, by which they do manage to considerably reduce the number of successful exploits. However, many advanced attacks still succeed to bypass such sandboxing [125]. Classical HIPS/HIDS (Host Intrusion Prevention / Detection Systems) technologies, like Comodo, hook certain APIs in order to detect malicious calls to those functions. The bypass method is usually trivial, for ex. by restoring the original code then calling directly the kernel or by calling other undocumented and unhooked APIs to perform the same functionality. Some other exploit shellcode detection solutions are based pattern-matching or statistical analysis of possible shellcode varies. Such protection attempts are bypassed by malware using ASCII-encoded shellcodes, obfuscation or polymorphic/metamorphic shellcode [84]. There are also other attempts to prevent exploit execution or code injection, like per-process instruction-set randomization [57]. However, until now such attempts proved to be theoretical, being very difficult to apply it in actual CPUs.

2.3. Brief survey of other attack techniques. Assumptions

Although not directly linked to client side protection, to have a broader overview of the context clients are working in, we need to consider some other attacks and related assumptions.

Man-in-the-middle attacks (MiM) focus on intercepting the communication between the server and the client. There are two generic ways such attacks are carried out. Many attacks try to brute force the captured network packets or try to break the encryption method using advanced mathematics. We do not cover this topic and assume that 256 bit SSL/AES cryptography to be safe. However, another type of MiM attacks (including most of the known SSL attacks) rely on breaking third-party Certificate Authorities [32, 93, 43]. Here we observe a very disturbing statistics, with Comodo, VeriSign and many other CAs being repeatedly hacked, leading to a point where many consider the usage of third-party CAs to be broken. We think that in any client-server scenarios that permit it (*e.g.* online banking or military), the server side shall issue their own certificates and become their own CA. This is easily doable in scenarios where a physical contact before system setup takes place between client and server side operators. From security point of view, this means, that hacking the CA equals directly hacking the server side. This we think is a strong enough assurance.

We can't talk about securing the client side of a network communication without considering phishing, widely used today to steal credentials [55]. Over 150 million phishing emails are sent daily, and around 80,000 persons become victims of those attacks each day [140]. However, phishing is just one example of well-crafted social engineering, and, while several protective measures can be taken, no one can protect in a

bulletproof way against users that willingly and knowingly give over credentials and their identity. Depending on the exact scenario, many protective approaches can be taken. For example, a dedicated online banking software can use built-in hardcoded URLs (thus avoiding any e-mail based phishing attempt). Other measures include the good practice to avoid shortened URLs or the routing of network traffic through a security proxy server which seeks to filter out all malicious content.

3. HARDWARE VIRTUALIZATION AND SECURITY

3.1. Bounding technology and security concepts

When designing a software-only security solution, in general, or a secure communication software, like an online banking application, in particular, for a typical computing system, one shall note that the OS, the applications running above, the security solution and any potential malware or cyberattack basically share the same computing resources: the same memory, the same CPUs and the same privilege levels (user-mode and kernel-mode). The concepts of different privilege levels or isolation are not new at all, nor is their implementation at hardware level. A notorious example for not employing the principle of isolation could be the fact, that although Intel introduced in 1985 the 80836 microprocessors, supporting four different levels of protection (rings 0, 1, 2 and 3), offering the possibility to isolate the kernel from the drivers, the executive and the applications into different, hardware enforced privilege zones, no widely deployed operating system like Windows or Linux ever used this [50].

Intel released VT-x hardware virtualization support in 2005, followed in 2006 by the release of VT-d and TXT technologies and in 2008 of VT-x/EPT extension [50, 48, 47]. In contrast with this, until this day, mainstream operating systems and security solution are using few to none of the possibilities of those technologies to strengthen security. The reasons behind this are many. Among them was the desire to maintain compatibility with older hardware and software, the lack of know-how and expertise, an industry wide inertia. One notable exception is Linux, where certain flavours support TXT for attestation of the OS kernel, but not to attest security solutions or applications within. Although VT-x, VT-x/EPT and VT-d are primarily focused to provide virtualization capabilities, *e.g.* to be able to run multiple operating systems as guests on a single physical hardware, they also have tremendous potential in the field of security. On the other hand, Intel TXT was specifically designed for security purposes, to allow strong cryptographic load time attestation of system software [35].

We shall also mention other technology elements that are strongly related to providing secure client environments. Second and third generation Intel Core i5/i7 CPUs include [50] built-in AES encryption support (AES-NI). Third generation Intel Core CPUs include built-in hardware random number generators. TPM chips include support for SHA-1 hash based binary image measurements, RSA cryptography and sealed storage [112].

Intel VT-x provides basic CPU virtualization capabilities. VT-x introduces a completely new level (ring -1 or VMX Root Mode) of execution privilege, more privileged than ring 0 used by the operating system kernel. This ring -1 level can also be used by security software – or a secure client side application, if we implement it as a virtual machine monitor (VMM), also known as a hypervisor. This basically provides a security software with the capability to intercept and control the execution of the operating system kernel, with the OS and all applications being moved into a virtual machine (VM), also known as a guest. Intel VT-x/EPT provides the capability to completely virtualize the physical memory space of a guest virtual machine and thus, to be able to control at a page level for each guest the read/write/execute access rights associated with memory [50]. We must also consider the fact that, by the design of the CPUs, there can be only one piece of software running at ring -1 at any time, so, as long as a first software controls the CPU from hypervisor level, no other – possibly malicious – software can get into ring -1.

Intel VT-d provides the capability to virtualize the access to the system memory for extension boards, devices and controllers connected through the system bus. Numerous devices like network cards, storage controllers etc. have built-in DMA controllers that give them direct access to the physical memory of the system using, bypassing any control normally imposed by the CPU. Employing the capabilities offered by Intel VT-d, a security solution is able to provide hardware enforced protection against DMA attacks [48].

Intel TXT is a feature that, based on Intel VT-x/EPT, VT-d and a TPM (Trusted Platform Module) chip [113] provides the system with the capability to load and launch a VMM software in a measured well know state that is cryptographically attested to be tamper-free [35, 47]. The challenges around system software attestation are widely known [65], with several attempts being made to enhance the integrity or attestability of computing platforms and system software (*e.g.* UEFI Secure Boot [117] or Measure Boot for Windows 8 [78]). The need and desire to have some kind of attestations is obvious: one doesn't really want to just know that "*a client side application has been installed on a system*" but would rather like to be ensured at least that "*the client side application that has been installed on the system was started / is running without being tampered with*".

3.2. Hypervisor based security and integrity attestation state-of-the-art

The academia researched heavily both the topic of integrity attestation and that of the employment of hypervisors to enforce security.

Hypervisors provide an excellent way to perform security monitoring from below the OS, running at a different privilege level. Garfinkel *et al.* [31] present an IDS (Intrusion Detection System) built using a hypervisor, Livewire, capable to extract data from the guest, then to reconstruct semantics of those data. Srivastava [98] presents a hypervisor-based architecture that can identify malicious code inside infected systems at process-level granularity. The identification is based on network-level security software, linking each malicious network activity with the process behind. Wang *et al.* introduce HookSafe [121], a hypervisor focused on prevention of kernel rootkit infections by protecting in guest kernel code against hooks. Chen *et al.* present a different approach with Overshadow [15], protecting the integrity of application data against compromised OS. This is achieved by presenting applications with a normal view of its resources, but the OS with an encrypted view. Seshadri *et al.* propose SecVisor [94], a tiny hypervisor aimed to ensure code integrity for commodity OS kernels, protecting them against kernel malware and zero day exploits.

Lampson [61] proposes creating different, well isolated VMs, according to different roles: 'red' VMs for generic applications, possibly compromised by malicious code and 'green' VMs for a critical, trusted applications handling sensitive data. Garfinkel *et al.* propose a similar approach introducing Terra [30], having remotely attestable VMs for specific applications, like DRM media players. Our proposed approach follows the isolation principles presented by Lampson and Garfinkel, respectively.

McCauley introduces a novel way with HyperPass [75] to protect user passwords from being found by malicious software. With HyperPass, the passwords are not stored anymore inside the guest system at all, but instead, are stored inside the hypervisor. HyperPass has complete control over the physical network interface, effectively filtering all network traffic. TLS encryption is handled with a legitimate man-in-the-middle approach. The key idea is, that passwords are filled-in into forms (e.g. at an online banking login) only from inside the hypervisor. This way they achieve a solid credential protection.

Martignoni *et al.* [71] introduce Cloud Terminal, proposing a split approach for security: a thin terminal running on the clients ensuring only communication with a remote, in-cloud rendering engine at the server side. The server side comprises a massive VMM supporting one VM per each client to effectively render application content. Their solution has resemblance with our proposal, as they support on-the-fly loading below a running instance of the host OS and remote attestation.

McCune *et al.* [76] present a novel way to execute security-sensitive code in complete isolation, named Flicker. This solution provides fine-grained attestation of the executed code to a remote party. However they have many limitation, including a severe performance penalty and the requirement to custom compile any sensitive code. McCune later presented TrustVisor [77], featuring much better performance with a considerably bigger codebase. Not covering KVM device virtualization, their approach has limited applicability.

Lee-Thorp [65] provides a great overview of the challenges regarding trusted computing based on TPM chips. He argues that extending the TCG (Trusted Computing Group) architecture and reworking application designs are the most viable routes to making attestation viable in practice. Lyle *et al.* analyze the problems of remote service attestation [67], but from the client's point of view. They argue that critical services, such as healthcare services, electronic voting services, financial services, grid services, all need to be attestable: whenever a user is using one of these services, how can he know if the service is attested (not tampered by a malicious intervention)? Their approach however is vulnerable to MiB attacks. More than this, the integrity

of the client systems in not considered at all. Armknecht presents [1] a pure software-based attestation framework. While a software attestation may increase the security of a system, it is far from providing the same level of security as hardware-assisted integrity attestation does. Huang *et al.* [42] present a practical approach of doing remote-attestation, using hardware-based attestation mechanisms. A protocol for remote attestation is proposed, which has the capability of verifying the integrity of an application with regard to a remote party. The protocol may, however, be the subject of various attacks known in the art.

To be fair in our analysis, we acknowledge that there are also researchers raising concerns regarding hypervisor based security. Heiser *et al.* [39] argue that once the kernel of an operating system can and has been formally verified, new possibilities for security arise. They are against the usage of hypervisors advocating that a verified, and thus really trustworthy, OS kernel can provide better security than hardware-virtualisation based strong isolation. While we admit the exceptional value provided by formally verified code, we strongly believe that the software industry is by far not ready to adopt formal verifications on a wide scale, not affording the increased costs and having neither the required tools nor the expertise.

Heiser *et al.* [145] also argues that there is actually a convergence between hypervisor and microkernel architectures. There are numerous security benefits in hypervisors built using a microkernel-like design approach. Steinberg *et al.* presents Nova [146], introducing a novel way to build a 'microvisor' with a trusted computing base at least of an order of magnitude smaller than common hypervisors. They use microkernel architecture for the hypervisor to achieve lower attack surface and better security.

We close this short review with a significant report. Zaharia *et al.* performed an in-depth analysis of previous academic research about hypervisor supported security [131], concluding: "*the goal* […] *is not to prevent computers from being compromised; the complexity of modern OSes and the gullibility of most users makes compromise too easy to reliably prevent. The goal, instead, is to allow compromised machines to still perform some useful tasks without sacrificing user information* […] *Hypervisors are well equipped for this*". We consider this a truly precise characterization of the situation we are dealing with.

We underline again, the sharp contrast between what has been done in academic and military research versus what has been applied widely in real-life practice. The ideas of creating reliable and secure systems, or to build security based on virtualization are not new at all. Between many other examples [29, 20, 38, 96], there is one very significant one: Karger *et al.* [53] spent a decade long effort between 1981 and 1990 focused on the development of a production-quality VMM security kernel capable of receiving an A1 (highest) rating from U.S. National Computer Society Centre. In spite of this, our feeling is that pretty much the whole history of personal computing, dominated by MS-DOS, Windows (and later Linux or Mac OS) systems, repeatedly and systematically sacrificed security for the sake of performance or compatibility with older software, among other reasons. As a historical example, and before any Linux fan would yell towards us, we would like to gently remind that MINIX was here before Linux, with a much more security supporting microkernel architecture [111]. We are however afraid, that there are still many users, marketing people and corporate leaders, that would sacrifice security for 5-10% performance gain. We found to be very significant M.E. Lesk's depiction [62] "*Right now, most people are still wondering 'what is cybersecurity?' As time goes on, we predict people will start to demand it; then want it cheaper, and then eventually forget it*".

On the field of hardware virtualization, there are few client-oriented type 1 bare metal products available. The most significant of them is Citrix's Xen Client XT [95], a security hardened client virtualization product. It uses all available Intel virtualization oriented technologies (VT-x/EPT, VT-d, TXT) to provide strong security, integrating also several introspection and anti-malware technologies. However, it is not only close sourced, but also restricted as a market only for the U.S. government and federal agencies. We are not aware of Xen Client XT being integrated with remote attestation servers, but the possibility of doing so is certainly in the reach of the vendor. We must also stress, that its deployment is highly restricted because it needs pre-installation, and thus, it is not an option for millions of existing users out there.

Intel / McAffee Deep Defender [73] is the only known product from a traditional anti-virus vendor that incorporates VT-x/EPT technologies, thus provides enhanced protection against rootkits. However, it has several severe limitations and drawbacks, as it does not employ VT-d [74], thus being vulnerable to DMA attacks, and does not employ Intel TXT to assess the solution's integrity.

Qubes OS [86] is likely the most noticeable research project focused to deliver a heavily security oriented environment, based on the open source Xen hypervisor and involving all Intel security features (VT-x/EPT, VT-d and TXT). It is the best example for security-by-isolation, however, involving also some drawbacks. First of all, it does not employ any kind of anti-malware protection, trying just to encapsulate /

isolate any exploit or damage to a specified perimeter (working zone). Secondly, they have no support for remote attestation. We recognise that they could overcome their limitations and we think it would be great to see much more Qubes-like approaches and solutions in widely deployed, real-life applications.

Bromium Microvisor / vSentry [12] aims to provide enhanced security based on per-process virtualization. They do not offer trial versions publicly for detailed evaluation, but according to the available information they focus to secure major applications, including Microsoft Office or Adobe Reader by strong isolation and sandboxing. They do not employ Intel VT-d or Intel TXT and thus their approach is both vulnerable to DMA attacks and lacks integrity attestation support. Another critical aspect is that they run more likely a type 2 hypervisor, based on the environment provided by the host OS and not as a type 1 bare metal hypervisor, thus potentially exposing a much greater attack surface.

4. TOWARDS TRUSTED CLIENTS

4.1. Proposed general architecture

Bearing in mind the failure of today's software-only security solutions, it comes naturally for us to build our proposed design around a security tailored, client oriented, hardware virtualization based type 1 bare-metal hypervisor. We choose this solution because hypervisors numerous advantages, including strong insolation, a significantly reduced codebase – thus a small attack surface and the possibility to analyse or enforce security from outside the operating system. We took this approach considering also the red-green separation principles described by Lampson [61].

We use the hypervisor to take complete control over critical hardware devices and isolate the secure client application inside a trusted VM running heavily customized Linux. The hypervisor employs all known VT-x/EPT, VT-d and TXT security technologies. Our approach, as illustrated in Fig. 1, comprises two different operating environments, isolated by the hypervisor.

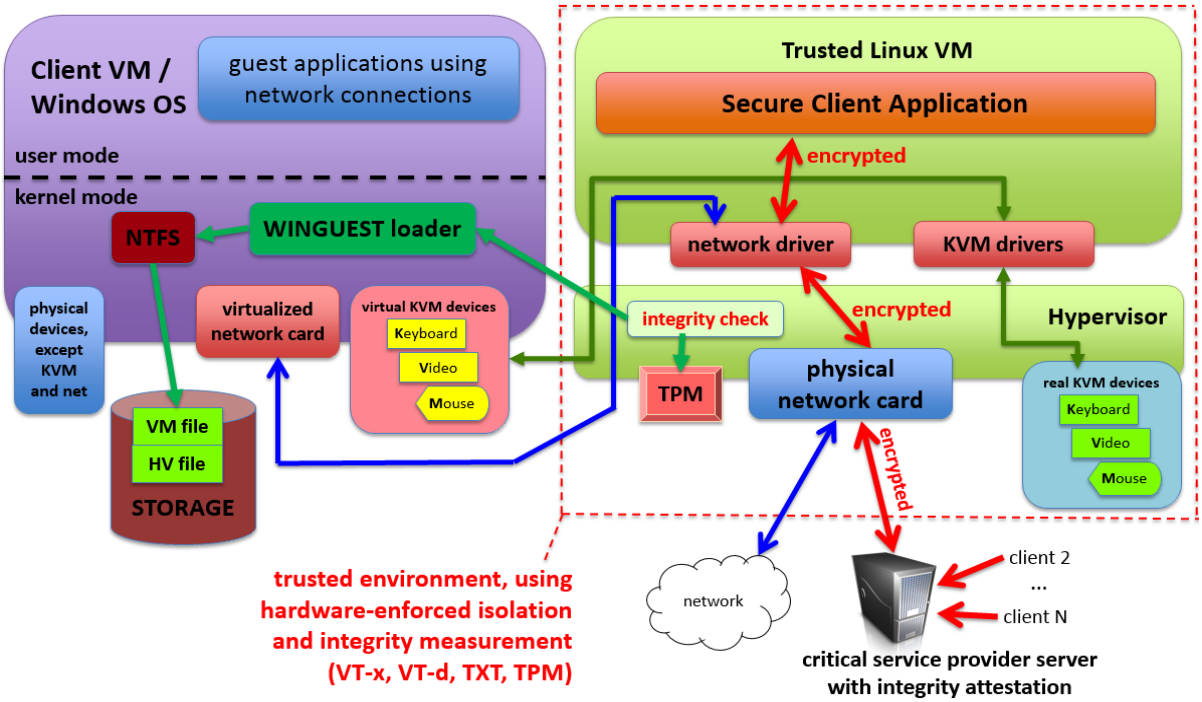


Fig. 1 – High level overview of key components of our proposal. The hypervisor completely takes over the physical KVM devices, thus ensuring any malware targeting our OS cannot interfere with the client application or its network communication. All critical communication is done through encrypted channels. The hypervisor, the Linux VM and the secure client application are hardware-isolated from OS, their integrity is enforced by Intel TXT and TPM. Both the hypervisor and the Linux VM are loaded from monolithic files (not requiring a separate partition), with their integrity being measured load-time, then validated.

The first environment is the existing client Windows operating system, inside which we install a kernel driver (winguest). Its main role is to load and start the trusted environment, for example in response to a hotkey combination hit by the user. It is important to note that we do not assume or require the client VM to be malware-free. For example, it can be infected with advanced malware designed to steal sensitive data, including kernel-mode rootkits. As we will outline in Section 4.4, such a malware can at most perform a DoS (Denial-of-Service) attack against us: it can prevent the loading of the trusted environment, but if the loading succeeds, we are assured that the client is in a cryptographically measured, tamper-free state. Once the trusted environment is loaded, the client VM and operating system will lose direct control of several physical devices, including not only processor, memory and chipset, but also network cards and KVM (keyboard, video and mouse) devices. In order to allow the client VM to fluently continue its execution, the hypervisor will expose to the OS virtualized versions of the devices described above, that will simulate the presence of the physical devices.

The second environment comprises a custom Linux VM running a secure client application, which can be specific to the exact deployment scenario (*e.g.* a security hardened online banking client). The integrity measurements based on Intel TXT and the TPM chip cover not only the hypervisor, but also the trusted VM, the included real device drivers and the secure client application also. The Intel TXT mechanisms ensure us, that if the environment is loaded, then (i) it can completely take over the control of critical CPU, chipset and memory devices, and (ii) the loading was performed in a tamper-free manner. The hypervisor will allow Linux to directly control the physical devices that have been hidden from Windows. This approach will allow the creation of communication channels between the virtualized devices exposed to Windows and the physical devices.

Both operating environments (VMs) are assured shared access to physical processor and to I/O devices in order to continue their execution fluently. The hiding of the physical devices from the client OS is achieved by taking over the complete communication interface between the drivers and the underlying devices, *i.e.* by rendering the device's slot as empty on the PCI bus, then triggering a hardware rescan in the client OS. In a similar way, the simulated hardware appears as newly inserted (hot plugged) devices, for which, in turn, Windows fluently loads new drivers and I/O stacks.

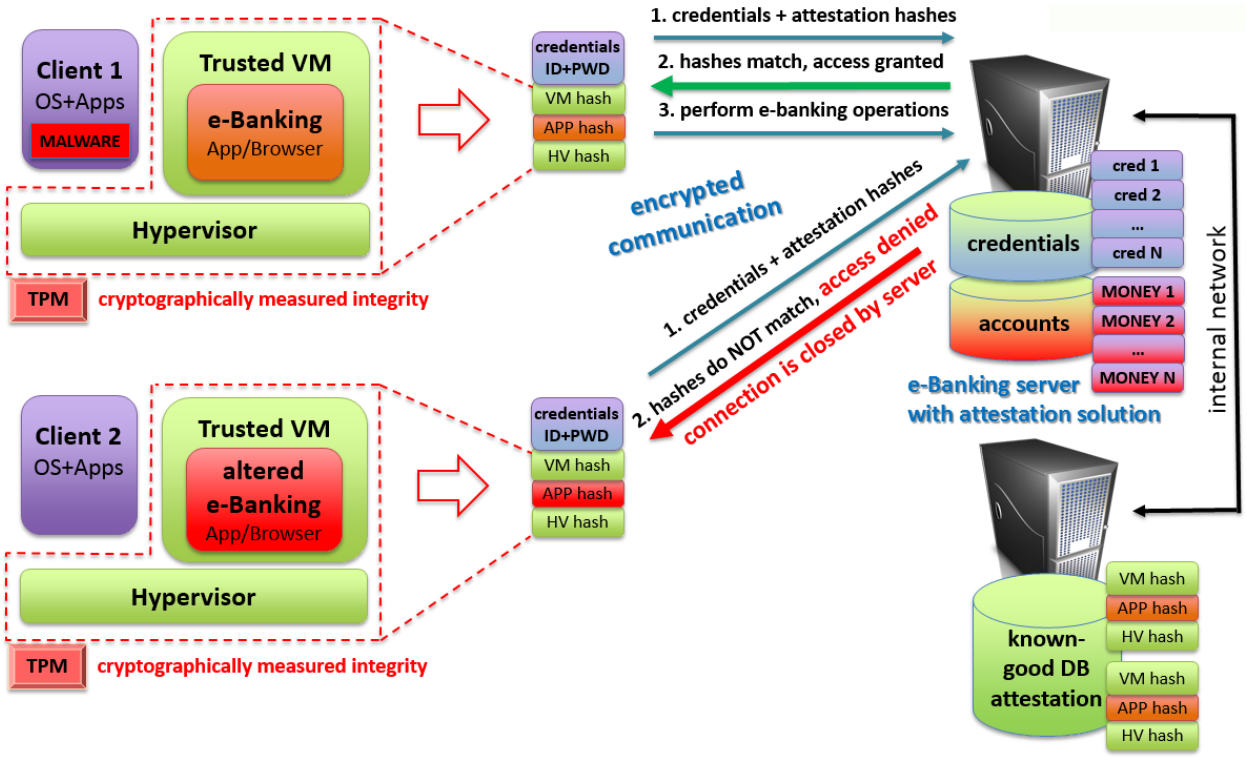


Fig. 2 – One possible application, comprising hypervisor based online banking application. The server side includes an integrity attestation server, holding all known-to-be-good integrity hashes in a database.

To better illustrate the proposed solution, we present one possible application in Fig. 2, depicting two online banking clients trying to connect to a service provider server from the bank. The bank also performs remote integrity attestation of the clients, requiring not only valid login credentials but known-to-be-good client integrity measurement hashes. The first client successfully connects even if the operating system or its applications are altered by a targeted attack or an advanced malware, because the e-Banking VM, the hypervisor and the e-Banking application have tamper-free, cryptographically attested integrity. In contrast with this, the connection of the second client, which is tampered with, is denied, because the e-Banking application's hash does not match the one in the known-good hash database. We expressly choose to present this with external clients having no direct connection at all to the attestation server: this way, in order for an external attack to reach the attestation server, it would require to go through existing vital banking infrastructure. In a way, we can consider the integrity of the known-to-be-good attestation database to be at least as valuable as money itself.

4.2. Particularities and strengths

The proposed approach promises hardware enforced strong protection against various types of attacks and malicious code affecting the client. Our solution allows us to perform secure business with a client, even if the endpoint is infected by advanced malware, including kernel rootkits, password stealers or Trojans.

An advantage of our approach is the inclusion of a full hypervisor and a complete Linux VM in two-three key files (*e.g.* the hypervisor binary, an in-guest driver and a VM image), allowing easy deployment on a client, without the need for end-users to reimage their systems. A related important aspect is, that we can install it under an already existing operating system, which can be already compromised by malware.

Our hypervisor does not need to be started before the operating system, but is started on demand, just like an application, thus greatly improving usability. An important strength is the ability to take over and virtualize runtime, on-the-fly from main operating system key hardware devices, *e.g.* network cards.

One more advantage of our proposal, although relevant for only very specific working scenarios, is the fact that we aim to completely take over the graphics card and assign it to the trusted VM. This way, secure clients can benefit from complete hardware accelerated graphics.

Finally, our proposal integrates well with existing anti-malware solutions, thus a finalized product based on it have the potential to reach tens of millions of client deployment easily.

4.3. Current progress, proof-of-concept results

Since 2011, anti-malware products from Bitdefender include a proprietary thin layer hypervisor as part of Active Virus Control technology. One critical property of it is the support of runtime on-the-fly loading from a single file stored inside the OS, hence it is very easily deployed with standard installation procedures. However, it does not incorporate Intel VT-d or Intel TXT technologies, nor complete interrupt virtualization.

Based on this initial work we have already done exploratory research to prove the feasibility of key ideas from our proposal. We have developed several proof-of-concept modules to demonstrate the feasibility of runtime on-the-fly VT-x/EPT and VT-d initialization, a minimal MLE (Measured Launch Environment) for Intel TXT integrity validations and for on-the-fly takeover of Ethernet network cards. We also demonstrated in-lab also the incorporation of encrypted network communication.

Our current proof-of-concept implementation has been demonstrated on four off-the-shelf, commonly available hardware systems, running Windows 7 and Windows 8 guest operating systems. We have tested several different Ethernet cards to validate multiple drivers from the Linux VM.

While it is too early to provide reliable numbers and statistics, we can mention several preliminary metrics. On-the-fly loading time of our hypervisor is around 3 seconds, including the time needed to bring up the Linux VM. The network virtualization has around 200 Mbit/s throughput without optimizations, deemed to be more than enough for the targeted client scenarios. The total research codebase is around 75 KLOC, excluding the customized Linux kernel.

We have not yet performed KVM device takeover, but considering our experience with Ethernet cards we estimate that to be technically feasible. The Intel TXT based integrity attestation proof-of-concept has been demonstrated on client level, without including any server side part.

4.4. Limitations. Possible advanced attack scenarios

Being focused on applied research, we must consider deployability and usability. Here we find a huge gap between various target audiences: although some of the key technological elements can be the same in home user, corporate and military scenarios, there are significant differences also. Home users would never accept to deploy on a massive scale a technology that needs reimaging from zero their system. They need easy and fast ways for deployment and very simple utilisation. At the other end of the spectrum, military needs maximum security, minimizing and mitigating every conceivable risks and attack vectors. We must be keep in mind those two extremes while analysing the strength and weaknesses of our proposal. We must stress, that most of the following limitations and possible attack scenarios do apply to existing commercial client security solutions and the vast majority of published research work, especially from the academia area. In spite of this, such aspects are usually left out or disconsidered, often generating a biased image.

We identified three important limitations of our proposal. First of all, in the current form we have no data persistence. Users working in the secure environment have no means of saving locally their files and any critical data. As we outline in Section 5.2, we aim to do further exploratory research on this topic. Secondly, the integrity attestation must include hashes covering the measurement of firmware code also, like BIOS / UEFI or SMM code for example. There is an inherent limitation regarding firmware updates, which can render the hypervisor and the trusted VM non-loadable due to hash changes. Thirdly, when switching to the trusted environment, either due to inherent differences between the physical and the emulated devices, or because of loss in the intermediate state of the physical devices and drivers, some existing applications might fail to continue without interruption until the trusted environment is closed, *e.g.* consider a game running on the client, dependent on the graphics adapter, or an application missing several network data packets.

There are numerous further attack techniques and scenarios to consider, while we are evaluating the security of a client. The following list is not exhaustive, but shall nevertheless be a solid starting point for any meticulous evaluation. We skip the ones already mentioned in Section 2.3 and focus only on client side attacks that can still, at least theoretically, affect clients protected by an implementation of our proposal.

Because we use runtime on-the-fly loading for the hypervisor instead of loading it before the operating systems starts, a kernel malware can attack our loader component (winguest). This way, they can perform a denial-of-service attack against us. We stress that such attacks can prevent the starting of our solution, but cannot steal our credentials or access confidential data. The solution is either loaded completely, with its integrity validated, or will fail to load at all.

Client systems can be physically stolen (actually, this is noticeably frequent even for government laptops). In such cases an attacker must break existing security measures (BIOS password, system password, hard disk password or encryption) before can try to launch the trusted environment. If he succeeds, he must brute force our credentials in order to retrieve any TPM sealed-in secrets. On the other hand, if he tries to connect to the server (we don't have local persistence for critical data), the server requires valid credentials, not only attested integrity, having the option to deny access after a few unsuccessful attempts.

Vulnerabilities in the OS kernel software network stack can lead to the compromise of the trusted VM, if an attacker can deliver an exploit, *e.g.* by a malformed network packet. This can be mitigated by isolating further the network drivers and the complete network stack into a third dedicated virtual machine. In this way, a successful attack doesn't end with a compromise of our credentials or critical data, but still allows the attacker to perform MiM or DoS scenarios. This approach is well illustrated by Qubes for example.

Although out of our control, especially in critical applications we shall not disconsider possible bugs and vulnerabilities in the CPU and / or built-in hardware virtualization technologies. Modern processors have numerous known small bugs and glitches. Some of them are fixed by microcode updates, and most of them carry no security impact at all. However, occasionally significant bugs are found that can allow an attacker to perform privilege escalations or to bypass critical security technologies. Wojtczuk *et al.* demonstrated [128] several software attacks against VT-d and against Intel TXT also [126]. There have been disclosed vulnerabilities that allow malware to crash the system [134] or vulnerabilities [138, 23] that allow a local authenticated attacker to perform operating system privilege escalation or a guest-to-host virtual machine escape.

Vulnerabilities in system firmware, like SMM or UEFI code can also lead to compromised clients. This has been demonstrated by Wojtczuk [127] and Duflot [21]. To mitigate such attacks one needs to virtualize also SMM mode, however today this is fully supported on AMD platforms only.

One of the key advantages of a hypervisor is the very low exposed attack surface. This however does not mean that a small attack surface (*e.g.* hypervisor ⇔ VM interface) is necessarily bulletproof. There have been several successful guest-to-host escape vulnerabilities demonstrated on commercial hypervisors [40]. Risks can be minimized by thoroughly reviewing and validating the interface. We also stress again, that this attack surface is significantly smaller than the attack surface exposed by conventional OS and applications.

Vulnerabilities in device firmware (*e.g.* network cards integrating vPRO and Intel AMT, but not only) have been underlined to be highly dangerous by Tereshkin [115] and Duflot [22], allowing remote, silent and practically undetectable compromise of systems. One mitigation of such attacks can be the isolation of the network cards inside a separate network dedicated virtual machine. Another mitigation can be the usage of non-vPRO/AMT network cards along with the proper refresh of the firmware from a trusted source.

Related to firmware based attack, one shall also evaluate the possibility of hardware backdoors, as exposed by Skorobogatov [101] and Brossard [10] among others. While unlikely to be a frequent issue in commercial applications, there are no known generic and efficient mitigation measures.

During ACPI S0⇔S3 (sleep) and S0⇔S4 (hibernate) transitions [135], by the very design of the Intel CPUs and of the TXT architecture, a hypervisor must give over the complete control of the CPU and of the hardware devices (at wakeup, execution will resume in 16 bit legacy real mode). We consider this to be not only a very nasty limitation, but the exposal of a significant attack surface. A secure hypervisor shall either deny completely such sleep state transitions, or completely unload itself and scramble the content of physical memory before entering sleep states.

Cross CPU core cache related side channel attacks have been demonstrated by numerous researchers, including Zhang [132] and Bangerter [6]. Such attacks can be used to either steal encryption keys or to directly access confidential data. Partial mitigations for such attacks is the storage of the encryption keys inside the registers of a CPU core or to completely reserve a CPU core for the trusted VM. Alternatively we can completely suspend the execution of the host OS while the trusted environment is running.

Cold boot attack can be used to retrieve not only encryption keys, but also to directly retrieve critical data from memory, as demonstrated by Halderman [37]. A partial mitigation for this can be the storage of encryption keys in CPU registers only, but decrypted data still can be accessed if system power is cut off before leaving the chance for the hypervisor to scramble the memory (we shall note also, that Intel TXT foresees surprise reboot scenarios using a special in-chipset bit, that requests the firmware to scramble the whole system memory if the last before-reset TXT session was not successfully closed).

At least theoretically, we shall consider a scenario when at install or load time we are running already inside a virtualized environment, *e.g.* a hypervisor level rootkit, supporting nested virtualization above. We consider this extremely hard to be done, as all technological elements, including VT-x/EPT, VT-d, TXT and the TPM chip must be properly virtualized, not only the basic VT-x extensions. Among other, Rutkowska [89] demonstrated hypervisor level rootkits, without however a complete technology emulation. Even the latest commercial Citrix and VMware products are still providing nested virtualization that is not feature complete. There are numerous hypervisor level rootkit detection techniques known in the art, as described by Lawson [64], Lauradoux [63] or Zovi [133].

Care must be taken with device virtualization to clear or scramble all device buffers and caches (*e.g.* network ring buffers, video RAM) while closing a trusted VM, before the client's OS regains control of the hardware devices. Dunn described cases when images and data where successfully recovered [24] after confidential browsing sessions ended.

To avoid disabling Intel TXT or performing TPM takeover from within the operating system (there are system management tools, like Dell Client Configuration Toolkit [19], that allow such operations to be carried out by administrators or a malicious script), systems must be password protected. It is yet to be answered if and how easily could an advanced malware crack such protection in a silent way. Nevertheless, we consider that the inclusion of management features that allow scripts to reconfigure BIOS / UEFI settings from within the OS is wrong by design, being the sacrifice of security for the sake of manageability. Beside this, being at hypervisor level we can deny any illicit BIOS update or reconfiguration.

An advanced Trojan running on the client's system might try to perform meticulous social engineering attacks in order to persuade the user to give away its credentials. Such malware can try to imitate the UI and the user experience of loading and running our hypervisor and the trusted VM. We shall note, that such attacks are limited to credential theft only, as they would be denied access to confidential data by the server

due to the lack of valid attestation hashes. A mitigation for such attacks can be the inclusion at install time of an encrypted, user specific personal image, sealed down using the platform's TPM. Such images can be used to visually confirm the user the authenticity of the exposed interface.

We must also consider the security impact of accessing from the trusted environment of documents or applications containing exploits or malicious code. A client incorporating third-party software or consuming third-party data (*e.g.* opening a malicious PDF or browsing of a website showing exploit-containing advertising images) might be easily compromised. The source of such data can be numerous, including other clients, the server itself, an USB stick (if allowed), the Internet (if open browsing is allowed) and so on. Such attempts can be part of advanced multi-stage attacks. The mitigation of such risks can be various, according to the specific usage of the client and the nature of the data. For some cases, strong hypervisor enforced per-process isolation can be used, as done by Bromium for example [12]. In online banking scenarios we can rely on a single preloaded known-to-be-safe browser and built-in certificates and URLs. For highly critical applications, no more than 'old typewriter style' information reviewing shall be allowed (*e.g.* plain text files in a very simple viewer; similarly, only very basic e-mail systems can be truly safe).

Hardware keyboard loggers, electromagnetic sniffers are much more efficient and easily available than average person would presume. They have been showed by Barisani [7], Vuagnoux [119], Kuhn [60] among many others, to be very effective in stealing not only credentials but critical data also, like secret e-mail messages as we type them into the system. The capturing and reconstruction of video output based on electromagnetic emissions of modern LCD is also practical, as demonstrated by Backes [4] for example. While such attacks and information thefts are easily carried out by a skilled person (or teams specialized on espionage), the mitigation of them is particularly difficult in most scenarios. Some initial mitigation steps can be the usage of so called TEMPEST-proof clients and the avoidance of direct power cable or cooper based network connections, *e.g.* using only interchangeable batteries and optical network cabling. Offering mitigation for some of the related issues, Grawrock argues [35] for the usage of trusted USB peripherals. They include cryptographic processors inside each device and require appropriate key setup to ensure trusted input channels between the peripherals and the drivers running inside the system. Electromagnetic attacks stand as a proof for our strong believe, already stated in Section 1.2, that the strength of a security solution can be measured only relative to the time and resources needed to be invested into an attempt to break it.

There are many other aspects and best-practices that needs to be considered while trying to ensure the security of client system. However, to detail them would exceed the scope of this paper. We also need to acknowledge and consider also that there is a valid criticisms regarding privacy issues and concerns related to TPM [143]. One fundamental issues here is, that clients must trust the TPM chip manufacturer, because the private keys are sealed in at manufacturing time [90]. It would be quite easy for a manufacturer, forced or not by authorities, to retain a list with all private keys and to give over them to third parties. At least for military applications, there is an open possibility to manufacture their own TPMs.

5. CONCLUSIONS

5.1. The value of our proposal. Possible applications. Contributions

We are confident that our proposal can provide much greater security in numerous usage scenarios, avoiding identity or critical business data theft, preventing leakage of credentials and ultimately saving time and money for home users, big enterprises and government consumers also. We strongly believe that our approach can be used on wide variety of desktops and laptops, widely available today. A key advantage of our proposal is the ease of deployment. Being simple to install and integrating well with existing Bitdefender anti-malware solutions, we have the potential to reach tens of millions of clients. What we propose is not unprecedented in term of technology, but our approach can make a paradigm shift across widely deployed security solutions, delivering instead of traditional software-only methods a new, hardware-enforced security solution for the masses. We believe this to be a significant step ahead.

While theoretically not bulletproof, our proposal significantly raises the cost of an attack along with greatly reducing the attack surface, in comparison with nowadays widely deployed and used, mass market approaches. It has the strength to protect clients against an impressive range of advanced malware and attack

techniques widely used by cybercriminals. On top of this, it has the potential to provide guarantee for service providers not only of the identity, but also about the integrity of clients connecting to them over network.

One notable contribution of our paper is the brief technical review of a wide range of real-life attack techniques impacting the security and integrity of client systems, paired with another review of numerous limitations and remaining possible attack methods after our proposal is fully implemented. We feel that a considerable share of published research work has limited real applicability, in part because of the lack of a proper evaluation of state-of-the-art security attacks and their implications. There is a considerable difference between real-life, market and ideal laboratory realities.

We estimate the first complete implementation of our proposal to be a highly secure online banking application, having huge potential impact, including significant direct and indirect cost savings both for clients and financial institutions. According to a 2011 analysis and forecast done by the French National Gendarmerie [122], in this decade we are going to see mass theft of banking, financial, or card information, all exceeding many times in impact and cost classic criminality / burglary. We are already on the verge of it. A 2012 Symantec study [109] estimates, that during 2012 alone 556 million adults worldwide experienced some form of cybercrime. Given this real-life context, our proposal has vast inherent value and a significant market potential.

The possible practical applications of our proposal are by far not limited to secure online banking. A hypervisor and Intel TXT based solution can be the cornerstone for working environments that provide secure e-mail or secure video conferencing for example. Such a setup can be used to create trusted working partitions in BYOD scenarios, or to permit employees to remotely access and control critical infrastructure in a very secure way. We strongly believe that our approach has significant applicability in numerous military and government areas also.

5.2. Future research. Next steps

In the months to come, we aim to focus research on creating a feature complete secure online banking solution prototype. This shall be the first complete and full scale validation of our proposal. To do this, we still need to validate at proof-of-concept level the technical possibility to fluently and safely take over from the OS, on-the-fly, all KVM devices. A related research area we would like to explore comprises the ability to do non-concurrent sharing of hardware resources between a common and a trusted partition, using fast VM switching, similarly with the methods proposed by J. Sun *et al.* [147] and K. Sun *et al.* [148].

We would like to extend our research efforts with the inclusion of some mechanisms that permit limited persistence for user data (*e.g.* emails, documents received by the client in trusted VM). One way to achieve this is to save all data in encrypted form to the untrusted VM, then store for all files a corresponding hash in a meta-file, which in turn is encrypted and sealed into the TPM.

We also plan to do exploratory research and feasibility evaluation for several of the remaining attack strategies presented in Section 4.5, like cold boot or firmware based attacks. It will require a great effort to collect and summarize in all those attack directions the most reliable information and know-how about the latest advancements. However, we feel this is essential in order to maintain an edge over our adversaries.

In the mid future we plan to extend our research efforts to include also cryptographic evaluation and hardening of the communication channel and the integrity attestation mechanisms. For this, we are already in talks with an academic cryptography research team.

ACKNOWLEDGEMENTS

We are grateful to Bogdan Dumitru, Rareş Ştefan and Adrian Coleşa for their support, encouragement and valuable advice.

REFERENCES

1. F. Armknect, A.R. Sadeghi, S. Schulz, C. WAchsmann, *Towards Provably Secure Software Attestation*, Cryptology ePrint Archive, http://eprint.iacr.org/, 2013.
2. D. Aumaitre, C. Delvin, *Subverting Windows 7 x64 Kernel with DMA attacks*, Hack in the Box Conference, 2010.
3. T. Avgerinos, S.K. Cha, B. Lim, T. Hao, D. Brumley, *AEG: Automatic Exploit Generation*, 18th Annual Network & Distributed System Security Symposium, 6 February 2011.

4. M. Backes, M. Durmuth, D. Unruh, *Compromising Reflections – or – How to Read LCD Monitors Around the Corner*, IEEE Symposium on Security and Privacy, pp. 158-169, 18 May 2008.
5. G. Bakas, *Windows Kernel Vulnerability Research and Exploitation*, RUXCON Conference talk, 19 November 2011, http://2011.ruxcon.org.au/2011-talks/windows-kernel-vulnerability-research-and-exploitation/.
6. E. Bangerter, D. Gullasch, S. Krenn, *Cache Games – Bringing Access-Based Cache Attacks on AES to Practice*, Proceedings of the 2011 IEEE Symposium on Security and Privacy, pp.490-505, 2011.
7. A. Barisani, D. Bianco, *Sniffing Keystrokes with Lasers/Voltmeters*, Black Hat USA, 25 July 2009.
8. B. Bencsáth, G. Pék, L. Buttyán, M. Félegyházi, *Duqu: A Stuxnet-like malware found in the wild*, CrySyS Lab Technical Report, 14 October 2011, http://www.crysys.hu/publications/files/bencsathPBF11duqu.pdf.
9. Bitdefender, *Bitdefender Internet Security 2013*, http://www.bitdefender.com/solutions/internet-security.html.
10. J. Bossard, *Hardware backdooring is practical*, Black Hat Briefings and Defcon Conferences, 2012.
11. R. Breauk, A. Spruyt, *Integrating DMA attacks in exploitation frameworks*, 7 February 2012, http://staff.science.uva.nl/~delaat/rp/2011-2012/p14/report.pdf
12. Bromium, *Micro-virtualization vs. Software Sandboxing - A comparison*, Bromium whitepaper, 2013, http://www.bromium.com/sites/default/files/Bromium-Whitepaper-Micro-virtualization-vs-sandboxing.pdf
13. C. Cerrudo, Easy local Windows Kernel exploitation, Black Hat USA, 21 July 2012.
14. U.S. CERT – National Vulnerability Database, http://nvd.nist.gov/
15. X. Chen, T. Garfinkel, E.C. Lewis *et al.*, *Overshadow: A Virtualization-Based Approach to Retrofitting Protection in Commodity Operating Systems*, Proceedings of the 13th international conference on architectural support for programming languages and operating systems, pp. 2-13, 2008.
16. CSIS, *Significant Cyber Incidents Since 2006*, Center for Strategic and International Studies report, February 2013, http://csis.org/files/publication/130318_Significant_Cyber_Incidents_Since_2006.pdf
17. D. Dagon, P. Vixie, *AV Evasion Through Malicious Generative Programs*, https://malfease.oarci.net/help/av_evasion.pdf
18. O. Delgado, A. Fuster-Sabater, J.M. Sierra, *Analysis of new threats to online banking authentication schemes*, Actas de la X RECSI, Salamanca, 2008.
19. DELL, *Dell Client Configuration Toolkit – CCTK*, 2013, http://en.community.dell.com/techcenter/systems-management/w/wiki/1952.dell-client-configuration-toolkit-cctk.aspx
20. J.E. Dobson, B. Randell, *Building reliable secure computing systems out of unreliable insecure components*, Proceedings of the Conference on Security and Privacy, Oakland, USA, pp. 187-193, IEEE, 1986.
21. L. Duflot, O. Levillain, B. Morin, O. Grumelard, *Getting into the SMRAM: SMM Reloaded*, CanSecWest conference presentation, 2009.
22. L. Duflot, Y.A. Perez, B. Morin, *What if you can't trust your network card?*, Proceedings of the 14th international conference on Recent Advances in Intrusion Detection, pp. 378-397, 2011.
23. G. Dunlap, *The Intel SYSRET privilege escalation*, blog, 13 June 2012, http://blog.xen.org/index.php/2012/06/13/the-intel-sysret-privilege-escalation/.
24. A.M. Dunn, M.Z. Lee *et al.*, *Eternal Sunshine of the Spotless Machine: Protecting Privacy with Ephemeral Channels*, Proceedings of the 10th USENIX conference on Operating Systems Design and Implementation, pp. 61-75, 2012.
25. ENISA, *Flash note: EU cyber security agency ENISA; "High Roller" online bank robberies reveal security gaps*, ENISA press release, http://www.enisa.europa.eu/, 5 July 2012.
26. FBI, *Three Alleged International Cyber Criminals Responsible for Creating and Distributing Virus That Infected Over One Million Computers and Caused Tens of Millions of Dollars in Losses Charged in Manhattan Federal Court*, FBI New York Field Office press release, 23 January 2013.
27. D. Florencio, C. Herley, *Is Everything We Know About Password-Stealing Wrong?*, IEEE Security & Privacy, IEEE, Vol. **10**, Issue 6, pp. 63-69, November 2012.
28. E. Gadaix, *GSM and 3G Security*, Black Hat ASIA, April 2001.
29. M. Gasser, *Building a Secure Computing System*, Macmillan of Canada, 1988.
30. T. Garfinkel, B. Pfaff, J. Chow, M. Rosenblum, D. Boneh, *Terra: A Virtual Machine-Based Platform for Trusted Computing*, Proceedings of the 19th ACM symposium on Operating systems principles, pp. 193-206, December 2003.
31. T. Garfinkel, M. Rosenblum, *Virtual Machine Introspection Based Architecture for Intrusion Detection*, Proceeding of Network and Distributed Systems Security Symposium, 2003.
32. D. Goddin, *How is SSL hopelessly broken? Let us count the ways*, The Register news article, 11 April 2011.
33. D. Goddin, *Certificate stolen from Malaysian Gov. used to sign malware*, The Register news article, 14 November 2011.
34. J. Granneman, *Antivirus evasion techniques show ease in avoiding antivirus detection*, 2013, http://searchsecurity.techtarget.com/feature/Antivirus-evasion-techniques-show-ease-in-avoiding-antivirus-detection.
35. D. Grawrock, *Dynamics of a Trusted Platform*, Intel Press, 1st edition, 2009.
36. D. Gupta, X. Li, *Defeating PatchGuard – Bypassing Kernel Security Patch Protection in Microsoft Windows*, joint McAfee – Intel whitepaper, 2011, http://www.mcafee.com/us/resources/reports/rp-defeating-patchguard.pdf
37. J.A. Halderman, S.D. Schoen, N. Heninger *et al.*, *Lest We Remember: Cold Boot Attacks on Encryption Keys*, Communications of the ACM – Security in the Browser, Magazine, Vol. **52**, Issue 5, pp. 91-98, May 2009.
38. N. Hardy, *KeyKOS architecture*, ACM SIGOPS Operating Systems Review, Vol. **19**, Issue 4, pp. 8-25, October 1985.
39. G. Heiser, L. Ryzhyk, M. Von Tessin, A. Budzynowski, *What If You Could Actually Trust Your Kernel?*, Proceedings of the 13th USENIX conference on Hot topics in operating systems, pp. 27-32, 2011.
40. K.J. Higgins, *Hacking Tool Lets A VM Break Out And Attack Its Host*, http://www.darkreading.com/ article, 4 June 2009.
41. HP, *Cybercrime Costs Rise Nearly 40 Percent, Attack Frequency Doubles*, HP Research press release, 8 October 2012.

42. X. Huang, Y. Peng, *An Effective Approach for Remote Attestation in Trusted Computing* , Proceedings of the 2009 International Symposium on Web Information Systems and Applications, pp. 80-83, May 2009.
43. R. Hurst, *Is SSL Broken?*, blog, 3 February 2013, https://www.globalsign.com/blog/is-ssl-broken.html
44. IBM, *Rising Attacks Focus on Browsers and Social Media Networks*, IBM X-Force 2012 Mid-Year Trend and Risk Report, September 2012.
45. IDC, *IDC Financial Insights Consumer Payments Survey 2012*, July 2012, http://www.idc-fi.com/getdoc.jsp?containerId=prUS23585112
46. INTEL, *Intel Identity Protection Technology (Intel IPT)*, http://www.intel.com/content/www/us/en/architecture-and-technology/identity-protection/identity-protection-technology-general.html
47. INTEL, *Trusted Compute Pools with Intel Trusted Execution Technology (Intel TXT)*, http://www.intel.com/content/www/us/en/architecture-and-technology/trusted-execution-technology/malware-reduction-general-technology.html
48. INTEL, *Intel Virtualization Technology for Directed I/O, Architecture Specification*, http://www.intel.com/content/www/us/en/intelligent-systems/intel-technology/vt-directed-io-spec.html
49. INTEL, *Crimeware Protection: 3rd Generation Intel Core vPro Processors*, Intel Whitepaper, 2012, http://www.intel.com/content/dam/www/public/us/en/documents/white-papers/3rd-gen-core-vpro-security-paper.pdf
50. INTEL, *Intel 64 and IA-32 Architectures Software Developer Manuals*, http://www.intel.com/content/www/us/en/processors/architectures-software-developer-manuals.html
51. C. Isidore, *8 charged in $45 million cybertheft bank heist*, CNN Money article, 10 May 2013.
52. E. Kalige, D. Burkey, *A Case Study of Eurograbber: How 36 Million Euros was Stolen via Malware*, joint Verasafe –CheckPoint whitepaper, 2012, http://www.checkpoint.com/products/downloads/whitepapers/Eurograbber_White_Paper.pdf
53. P.A. Karger, M.E. Zurko, D.W. Bonin, A.H. Mason, C.E. Kahn, *A Retrospective on the VAX VMM Security Kernel*, IEEE Transactions on Software Engineering, Vol. **17**, No. 11, pp. 1147-1165, November 1991.
54. KASPERSKY, *Gauss: Nation-state cyber-surveillance meets banking Trojan*, Kaspersky Lab research blog, 9 August 2012, http://www.securelist.com/en/blog/208193767/
55. KASPERSKY, *Protecting your bank account using Safe Money technology*, Kaspersky Whitepaper, 2012. http://www.kaspersky.com/downloads/pdf/kaspersky_lab_whitepaper_safe_money_eng_final.pdf
56. KASPERSKY, *Duqu: Steal Everything*, Kaspersky Lab research, 27 March 2012, http://www.kaspersky.com/about/press/major_malware_outbreaks/duqu
57. G.S. KC, A.D. Keromytis, V. Prevelakis, *Countering Code-Injection Attacks With Instruction-Set Randomization*, Proceedings of the 10th ACM conference on Computer and communications security, pp. 272-280, 2003.
58. Kensigton, *15 remarkable remote working statistics,* blog, 7 June 2012, http://clicksafe.kensington.com/laptop-security-blog/bid/83929/15-remarkable-remote-working-statistics#axzz2VDhGJl49
59. B. Krebs, *Attackers Hit Weak Spots in 2-Factor Authentication*, security blog, 5 June 2012, http://krebsonsecurity.com/2012/06/attackers-target-weak-spots-in-2-factor-authentication/
60. M.G. Kuhn, R.J. Anderson, *Soft Tempest: Hidden Data Transmission Using Electromagnetic Emanations*, Proceedings of 2nd Workshop on Information Hiding, pp. 124-142, 1998.
61. B. Lampson, *Accountability and Freedom*, Microsoft Research, 26 September 2005, http://research.microsoft.com/en-us/um/people/blampson/slides/AccountabilityAndFreedom.pdf
62. C. Landwehr, D. Boneh, John C. Mitchell, S. M. Bellovin, S. Landau, M. E. Lesk, *Privacy and Cybersecurity: The Next 100 Years*, Proceedings of the IEEE, Vol. **100**, pp. 1659-1673, 13 May 2012.
63. C. Lauradoux, *Detecting Virtual Rootkits with Cover Channels*, Annual meeting of the European Institute for Computer Antivirus Research – EICAR, April 2008.
64. N. Lawson, *Don't Tell Joanna, The Virtualization Rootkit is Dead*, Black Hat USA, 28 July 2007.
65. A. Lee-Thorp, *Attestation in Trusted Computing: Challenges and Potential Solutions*, Technical Report RHUL-MA-2010-09, March 2010, https://www.ma.rhul.ac.uk/static/techrep/2010/RHUL-MA-2010-09.pdf
66. M. Lenon, *Microsoft Certificate Was Used to Sign 'Flame' Malware,* news article, 4 June 2012, http://www.securityweek.com/microsoft-unauthorized-certificate-was-used-sign-flame-malware
67. J. Lyle, *Trustworthy Services Through Attestation,* PhD Thesis, Keble College, University of Oxford, 2010.
68. MANDIANT, *APT1 – Exposing One of China's Cyber Espionage Units*, Mandiant Intelligence Center Report, 2013, http://intelreport.mandiant.com/Mandiant_APT1_Report.pdf
69. D. Marcus, T. Sawicki, *The New Reality of Stealth Crimeware*, joint McAfee – Intel whitepaper, 2011, http://www.mcafee.com/us/resources/white-papers/wp-reality-of-stealth-crimeware.pdf
70. D. Marcus, R. Sherstobitoff, *Dissecting Operation High Roller*, joint McAfee – Guardian Analytics whitepaper, 2012, http://www.mcafee.com/us/resources/reports/rp-operation-high-roller.pdf
71. L. Martignoni, P. Poosankam, M. Zaharia *et al.*, *Cloud Terminal: Secure Access to Sensitive Applications from Untrusted Systems*, Proceedings of the 2012 USENIX conference on Annual Technical Conference, pp. 14-14, 2012.
72. MCAFEE, *McAfee Threats Report: Second Quarter 2012*,http://www.mcafee.com/us/resources/reports/rp-quarterly-threat-q2-2012.pdf
73. MCAFEE, *McAfee Deep Defender*, product datasheet, 2012,http://www.mcafee.com/us/resources/data-sheets/ds-deep-defender.pdf
74. MCAFEE, *McAfee Deep Defender Technical Evaluation and Best Practices Guide*, https://kc.mcafee.com/resources/sites/MCAFEE/content/live/PRODUCT_DOCUMENTATION/23000/PD23874/en_US/Deep_Defender_Best_Practices_Guide_Aug_2012.pdf
75. J. McCauley, R. Mittal, *HyperPass: Hypervisor Based Password Security*, project report, University of California, Berkeley, 2012.

76. J. McCune, B. Parno, A. Perring, M. Reiter, H. Isozaki, *Flicker: an execution infrastructure for TCB minimization,* Proceedings of the 3rd ACM SIGOPS/EuroSys Conference on Computer Systems, pp. 315-328, 2008.
77. J. McCune, Y. Li, N. Qu, Z. Zhou, A. Datta, V. Gligor, A. Perrig, *TrustVisor: Efficient TCB Reduction and Attestation*, Proceedings of IEEE Symposium on Security and Privacy, pp. 143-158, 2010.
78. MICROSOFT, *Measured Boot*, MSDN technical documentation, http://msdn.microsoft.com/en-us/library/windows/desktop/hh848050(v=vs.85).aspx
79. E. Nakashima, *Confidential report lists U.S. weapons system designs compromised by Chinese cyberspies,* The Washington Post article, 28 May 2013.
80. Norman, *Security you can bank on*, Norman whitepaper, April 2013, download01.norman.no/whitepapers/shark/Financial-Services-White-Paper-security-You-Can-Bank-On-v1.0-Final.pdf
81. J. O'Dell, *How Much Does Identity Theft Cost?*, infographic, 29 January 2011, http://mashable.com/2011/01/28/identity-theft-infographic/
82. M. Paik, *Stragglers of the Herd Get Eaten: Security Concerns for GSM Mobile Banking Applications*, HotMobile'10 Proceedings of the 11th Workshop on Mobile Computing, Systems and Applications, pp. 54-59, 2010.
83. P. Passeri, *Comprehensive cyberattack reports / 2012 Cyber Attacks Statistics*, http://hackmageddon.com/, http://hackmageddon.com/2012-cyber-attacks-timeline-master-index/
84. M. Polychronakis, K.A. Anagnostakis, E.P. Markatos, *Comprehensive Shellcode Detection using Runtime Heuristics*, Proceedings of the 26th Annual Computer Security Applications Conference, pp. 287-296, 2010.
85. L.Y. Qing, *Cloud a haven for cybercriminals*, ZDNET article, 7 February, 2011.
86. Qubes, *Qubes OS Project homepage*, http://qubes-os.org/Home.html
87. A. Rassokhin, D. Oleksyuk, *TDSS botnet: full disclosure*, Esage Lab security analysis, 13 March 2012, http://nobunkum.ru/analytics/en-tdss-botnet
88. P. Rosenzweig, Significant Cyber Attacks on Federal Systems – 2004-present, blog, 7 May 2012, http://www.lawfareblog.com/2012/05/significant-cyber-attacks-on-federal-systems-2004-present/
89. J. Rutkowska, A. Tereshkin, *Bluepilling the Xen Hypervisor*, Black Hat USA, 7 August 2008.
90. J. Rutkowska, *Trusted Execution In Untrusted Cloud*, blog, 13 December 2011, http://theinvisiblethings.blogspot.ro/2011_12_01_archive.html
91. F. L. Sang, V. Nicomette, Y. Deswarte, *I/O Attacks in Intel-PC Architectures and Countermeasures*, 2011 SysSec Workshop, pp. 19-26, 6 July 2011.
92. B. Schneier, *Hacking Two-Factor Authentication,* blog, September 2009, http://www.schneier.com/blog/archives/2009/09/hacking_two-fac.html
93. B. Schneier, *VeriSign Hacked, Successfully and Repeatedly, in 2010*, blog, February 2012, http://www.schneier.com/blog/archives/2012/02/verisign_hacked.html
94. A. Seshadri, M. Luk, N. Qu, A. Perrig*, SecVisor: A Tiny Hypervisor to Provide Lifetime Kernel Code Integrity for Commodity OSes,* ACM SIG OPS Operating Systems Review, Vol. **41**, Issue 6, pp. 335–350, December 2007.
95. N. Shah, *Meet High-Security Demands with XenClient XT 3.1*, Citrix blog, 25 April 2013, http://blogs.citrix.com/2013/04/25/meet-high-security-demands-with-xenclient-xt-3-1-now-with-a-free-trial/
96. J.S. Shapiro, N. Hardy, *EROS: A Principle-Driven Operating System from the Ground Up*, IEEE Software, pp. 26-33, January/February 2002.
97. H. Shinotsuka, *Malware Authors Using New Techniques to Evade Automated Threat Analysis Systems*, Symantec research blog, 28 October 2012, http://www.symantec.com/connect/blogs/malware-authors-using-new-techniques-evade-automated-threat-analysis-systems
98. A. Shirastava, *Robust and secure monitoring and attribution of malicious behaviors*, PhD Thesis, Georgia Institute of Technology, August 2011.
99. Skape, Skywing, *Bypassing PatchGuard on Windows x64*, security report, 2006, http://uninformed.org/?v=3&a=3&t=sumry
100. J. Skinner, E. Metcalf, *Preventing Stealthy Threats: A Proactive approach from Intel and McAfee*, Intel IT Talk to an Expert Series presentation, 2011, http://www.intel.com/content/dam/www/public/us/en/documents/best-practices/talk-to-an-expert-stealthy-threats-presentation.pdf
101. S. Skorobogatov, C. Woods, *Breakthrough silicon scanning discovers backdoor in military chip*, Cryptographic Hardware and Embedded Systems Workshop (CHES), 9 September 2012.
102. Sophos, *Ukrainian and Russian police arrest banking Trojan masterminds*, news report, 9 April 2013, http://nakedsecurity.sophos.com/2013/04/09/ukrainian-and-russian-police-arrest-banking-trojan-masterminds/
103. M. Ståhlberg, *The Trojan Money Spinner*, F-SECURE, Virus Bulletin Conference, 2007.
104. P. Stewin, I. Bystrov, *Understanding DMA Malware*, DIMVA2012 Proceedings of the 9th Conference on Detection of Intrusions and Malware & Vulnerability Assessment, 26 July 2012.
105. Symantec, *Windows Rootkit Overview*, Symantec Security Response whitepaper, 2005, http://www.symantec.com/avcenter/reference/windows.rootkit.overview.pdf
106. SYMANTEC, *Symantec Introduces Game Changer for Strong Authentication Credentials*, press release, 9 February 2011.
107. SYMANTEC, *Norton Study Calculates Cost of Global Cybercrime: $114 Billion Annually*, press release, 7 September 2011.
108. SYMANTEC, *Banking Trojans*, Symantec white paper, August 2011.
109. SYMANTEC, *Consumer Cybercrime Estimated at $110 Billion Annually*, Symantec press release, 5 September 2012.
110. SYMANTEC, *Symantec Internet Security Threat Report*, Vol. **18**, April 2013.
111. A.S. Tanenbaum, A.S. Woodhull, *Operating Systems: Design and Implementation*, Prentice Hall, 1987.
112. TCG, *TPM Main Specification*, Version 1.2, 2011, http://www.trustedcomputinggroup.org/resources/tpm_main_specification

113. TCG, *Trusted Platform Module Authentication*, technical specifications, http://www.trustedcomputinggroup.org/solutions/authentication
114. R. Tehan, *Cybersecurity: Authoritative Reports and Resources*, U.S. Congressional Research Service report, 24 May 2013.
115. A. Tereshkin, R. Wojtczuk, *Introducing Ring -3 Rootkits*, Black Hat USA, 29 July 2009, 2009.
116. S. VOGL, *Secure Hypervisors*, faculty research report, 2009, http://www.sec.in.tum.de/assets/lehre/ss09/seminar_virtualisierung/Secure_Hypervisors_S-Vogl.pdf
117. UEFI, *UEFI 2.3.1 Specification*, 2012, http://www.uefi.org/specs/
118. UNODC, *Comprehensive Study on Cybercrime*, United Nations Office on Drugs and Crime report, February 2013, http://www.unodc.org/documents/organized-crime/UNODC_CCPCJ_EG.4_2013/CYBERCRIME_STUDY_210213.pdf
119. M. Vuganox, S. Pasini, *Compromising Electromagnetic Emanations of Wired and Wireless Keyboards*, Proceedings of the 18th conference on USENIX security symposium, pp. 1-16, 2009.
120. P. Wagenseil, *Banking Trojan cleans out your account — you won't even see it*, NBC news article, 20 June 2012, www.nbcnews.com/id/47896468/ns/technology_and_science-security/t/banking-trojan-cleans-out-your-account-you-wont-even-see-it/
121. Z. Wang, X. Jiang, W. Cui, P. Ning, *Countering Kernel Rootkits with Lightweight Hook Protection*, Proceedings of the 16th ACM conference on Computer and communications security, pp. 545-554, 2009.
122. M. Watin-Augouard *et al.*, *Prospective Analysis on Trends in Cybercrime from 2011 to 2020*, French National Gendarmerie report, 2012, http://www.mcafee.com/us/resources/white-papers/wp-trends-in-cybercrime-2011-2020.pdf
123. T. Wilson, *New TDSS/TDL4 malware infects 46 of Fortune 500 companies*, news article, 18 September 2012, http://www.darkreading.com/attacks-breaches/new-tdsstdl4-malware-infects-46-of-fortu/240007496
124. C. Wisniewski, *Exposing the Money Behind the Malware*, Sophos Security Trent report, June 2012, http://www.sophos.com/en-us/medialibrary/Gated%20Assets/white%20papers/sophosmoneybehindmalwarewpna.pdf
125. R. Wojtczuk, R. Kashyap, *The Sandbox Roulette: are you ready to gamble?*, Black Hat EU, 12 March 2013.
126. R. Wojtczuk, J. Rutkowska, *Attacking Intel Trusted Execution Technology*, Black Hat USA, 18 February 2009.
127. R. Wojtczuk, J. Rutkowska, *Attacking SMM Memory via Intel CPU Cache Poisoning*, Invisible Things Lab, 2009, http://invisiblethingslab.com/
128. R. Wojtczuk, J. Rutkowska, *Following the White Rabbit: Software attacks against Intel VT-d technology*, Invisible Things Lab, April 2011, http://invisiblethingslab.com/
129. M. Wood, *Want My Autograph? The Use and Abuse of Digital Signatures by Malware*, SPOHOS, VB Conference, 2010.
130. C. Wüeest, *Current state of online Banking Trojans*, ITU-Impact Cybersecurity Forum presentation, 2012.
131. M. Zaharia, S. Katti, C. Grier *et al.*, *Hypervisors as a Foothold for Personal Computer Security: An Agenda for the Research Community*, Technical Report No. UCB/EECS-2012-12, University of California at Berkeley, 13 January 2012.
132. Y. Zhang, A. Juels, M.K. Reiter, T. Ristenpart, *Cross-VM Side Channels and Their Use to Extract Private Keys*, Proceedings of the 19th ACM Conference on Computer and Communications Security, pp. 305-316, 16 October 2012.
133. D.A.D. Zovi, *Hardware Virtualization Rootkits*, Black Hat USA, 29 July 2006.
134. –, *Possible VT-x enabled Intel CPU Crash Vulnerability*, http://seclists.org/fulldisclosure/2010/Mar/550, 31 March 2010.
135. –, *Advanced Configuration & Power Interface Specification*, Rev. 5.0, http://www.acpi.info/spec50.htm, December 2011.
136. –, *Annual Fraud Indicator*, UK National Fraud Authority report, March 2012, https://www.gov.uk/government/uploads/system/uploads/attachment_data/file/118530/annual-fraud-indicator-2012.pdf
137. –, *sKyWIper (a.k.a. Flame a.k.a. Flamer): A complex malware for targeted attacks*, CrySyS Lab Technical Report, 31 May 2012, http://www.crysys.hu/skywiper/skywiper.pdf
138. –, *Vulnerability Note VU#649219: SYSRET 64-bit operating system privilege escalation vulnerability on Intel CPU hardware*, US CERT Vulnerability Database, http://www.kb.cert.org/vuls/id/649219, 12 June 2012.
139. –, *Bypassing Microsoft Windows ASLR with a little help by MS-Help*, 24 August 2012, http://www.greyhathacker.net/?p=585
140. –, *Phishing: How many take the bait?*, infographic, Get Cyber Safe campaign led by Public Safety Canada on behalf of the Government of Canada, 2013, http://www.getcybersafe.gc.ca/cnt/rsrcs/nfgrphcs/nfgrphcs-2012-10-11-eng.aspx
141. –, *Certified online banking Trojan in the wild*, The H Security news, 22 February 2013.
142. –, *Detecting Polymorphic Malware*, LAVASOFT research, http://www.lavasoft.com/mylavasoft/securitycenter/whitepapers/detecting-polymorphic-malware
143. –, *Trusted Computing: Criticism*, Wikipedia, http://en.wikipedia.org/wiki/Trusted_Computing#Criticism
144. –, *Symantec Validation and ID Protection Service*, http://www.symantec.com/verisign/vip-authentication-service
145. G. Heiser, B. Leslie, *The OKL4 Microvisor: Convergence Point of Microkernels and Hypervisors*, Proceedings of the first ACM Asia-pacific workshop on Workshop on systems, APSys '10, pp. 19-24, 2010.
146. U. Steinberg, B. Kauer, *NOVA: a microhypervisor-based secure virtualization architecture*, Proceedings of the 5th European conference on Computer systems, EuroSys '10, pp. 209-222, 2010.
147. J. Sun, D. Zhou, S. Longerbeam, *Supporting Multiple OSes with OS Switching*, Proceedings of USENIX Annual Technical Conference, pp. 357-362, 2007.
148. K. Sun, J. Wangy, F. Zhang, A. Stavrou, *SecureSwitch: BIOS-Assisted Isolation and Switch between Trusted and Untrusted Commodity OSes*, NDSS Symposium 2012.
149. O. Whitehouse, *GS and ASLR in Windows Vista*, Black Hat Washington DC, 2007.

Note: all links have been verified on 7 June 2013.

Received July 17, 2013

THE PUBLISHING HOUSE
OF THE ROMANIAN ACADEMY

PROCEEDINGS OF THE ROMANIAN ACADEMY, Series A,
Volume 14, Special Issue 2013, pp. 357–364

MAINTAINING THE CONFIDENTIALITY OF ARCHIVED DATA

Catalin LEORDEANU[*], Dragos CIOCAN[*], Valentin CRISTEA[*]

[*] Faculty of Automatic Control and Computers, University 'Politehnica' of Bucharest, Romania
Corresponding author: Catalin LEORDEANU, E-mail: catalin.leordeanu@cs.pub.ro

This paper proposes a novel data protection method, whether we are talking about data transmitted over the internet, or just archived and stored on a disk. The proposed method allows the owner of the data to store secret information inside and archive and also enables him a degree of deniability regarding its existence. This approach can be useful in situations when the owner of the data is constrained to reveal secret information. The proposed method enables the encryption of two different files which are then added to the same archive. Considering that one of the files contains secret information the user has the option of decrypting one of the files, while plausibly denying the existence of any other data inside the archive. This goal is achieved by adding random padding data which cannot be distinguished from encrypted data. As an additional protection method, after the encryption step, the data is split into multiple sections, which are scrambled according to a given key.

Key words: Confidentiality, Privacy, Cryptography, Plausible deniability.

1. INTRODUCTION

Protecting data against attacks is a very important subject which has been the focus of numerous research efforts. It is still a very active field, as researchers continue to improve data security. This goal can be seen as a combination of confidentiality, integrity, availability and authenticity.

The continual evolution of communication technologies, as well as the increase in the amount of data transmitted over the internet, require increased security. In this situation the main solution to ensure this degree of security involves cryptography techniques. This is a good solution to ensure the fact that an external attacker is not able to access the secure data. The choice of the encryption algorithm is also very important and we need to ensure that it is difficult to break. There are however situations when this is not sufficient, either from the fact that an attacker with sufficient time and computing power might break the encryption, or the owner might be constrained through other means to reveal the secret key. There is therefore a need for a mechanism which will enable the owner of the data to be able to plausibly deny the existence of any secret data in the archive. Without such a mechanism any encryption becomes useless if the holder of the secret key can be forced to reveal it.

One of the approaches used to achieve this goal is steganography. While encryption can be used to transform the data into something which cannot be understood, steganography can obscure the actual existence of the secret data. For example, one of the most common methods in steganography is called “LSB insertion” and it is used to hide secret messages into images. The priciple of this method is based on the modification of the least significant bit from the 24 bit reprezentation of each pixel, to form a secret message.Since only the least signinficant bit is modified the difference in the image will be minimal and the secret message will only be read by someone who knows of its existence.

The proposed solution combines steganography methods with encryption in order to hide a secret message inside and encrypted archive. Using this solution an attacker may decrypt the data inside the archive, without realizing that there is another hidden message.

This paper is organized as follows. Section 2 describes similar research efforts, with an emphasis on deniable encryption solutions. Section 3 describes the proposed solution and the structure of the resulting archive., Section 4 presents the algorithms which were used, as well as the operation which are performed in

order to ensure data confidentiality. Section 5 contains experimental results using the proposed solution and Section 6 concludes this paper and outlines directions for future research.

2. RELATED WORK

There are many research efforts which handle data confidentiality. One such solution is covered by the Deniable File Systems (DFS)[1]. This refers to file systems which can hide a small part of itself. It differs from classic Encrypted File Systems through the fact that the folders and files are not visible. [2] However, a disadvantage of the Deniable File Systems is the fact that it covers the entire partition or file system. It is not very useful if the user only wishes to hide a small secret message inside another file. StegFS [4] is also a file system which is able to hide the existence of different sets of data. It is actually an extenion of the Ext2fs file system for linux and the data can be encrypted as part of different security levels. An attacker would not be able to know if he has discovered the secret keys to all the security levels.

Another solution to ensure data confidentiality is TrueCrypt [3]. It falls into the category of Deniable File Systems, which we mentioned earlier. It is able to use different cryptographic algorithms, such as AES, Serpent, Twofish and others to encrypt in real-time an entire disk or a partition. It can also offer plausible deniability through the creation of a hidden disk partition, which is kept inside a visible partition. It has however a number of security vulnerabilities [3] and it is also difficult to configure.

A basic solution for deniable encryption is presented in [5]. It is however difficult to implement and use. Most other solutions have used the term “plausible deniability”, which brings it closer to steganography through the insertion of secret data inside encrypted files.

3. OVERVIEW OF THE CONFIDENTIALITY SOLUTION

The goal of the solution is to provide a way to maintain the confidentiality of archived data. The use of random encrypted data blocks is not new. In most algorithms the encrypted information is divided intro blocks of fixed size. For example, in the case of the AES algorithm, these blocks of data have 128 or 256 bits. Since the data that we need to encrypt is rarely of a size multiple of the block size, we are forced to use padding. The solution presented in this paper is based on the insertion of hidden data inside this padding when creating an encrypted archive.

The proposed solution is able to encrypt two different files, each with its own encryption key, and place them inside a single archive. One of the files can contain sensitive secret information, which the user is trying to hide. The other one may contain less sensitive information, which the user may reveal in order to obscure the existence of the first file. There is a compression phase when the two files are combined so as not to arrouse suspicions regarding the size of the encrypted output. Padding containing random data is also added to further obscure the real information. The difference in size of the final archive may therefore be blamed on the padding.

Let us consider the following scenario. A user following this solution may encrypt a file *f1* of 800KB, and another file *f2* of 200KB containing sensitive information. Considering a padding of approximately 20% the resulting encrypted file will have a size of 1200KB, without compression. If the user is forced to reveal the secret key to decrypt the data, he may only reveal the key which decrypts file *f1*, while blaming the difference in size on a larger padding than the one which was used. The existence of the second file, *f2*, therefore remains hidden.

The structure of the output file is shown in Fig.1. Upon analysis the archive will look like a single block of encrypted data. The choice of the AES cypher[6], using 128b or 256b will be sufficient to discourage brute force attacks.

The internal structure of the file can be divided into 3 useful parts: the header(1) containing identification data, offsets and encryption parameters, the padding(2) and the encrypted files(3). Another goal of the proposed solution is that the output should present itself as a single block of data, making it difficult for an attacker to distinguish its components from the padding.

Fig. 1 – Structure of the output file.

3.1. The header

The header is a very important part of the archive, since it contains information describing the way that the rest of the data should be interpreted. The header also contains some padding, as to make it difficult for an attacker to identify its elements. The total length of the header is 256B.

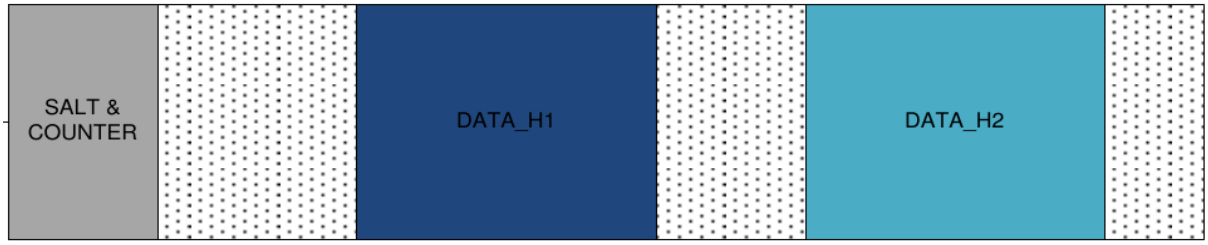


Fig. 2 – Structure of the header.

The header contains the following information:

– Salt & counter. The salt is a random generated number which will be added to the encryption key, to increase the difficulty of dictionary-based attacks. The counter, or Iteration counter also helps improve the security level of the encryption. These values are visible to an attacker and they are placed at the beginning of the header.

– Two data blocks Data_H1 and Data_H2. Each of these contains the offset and the size of the two encrypted files. Their positions inside the header are dependent on the encryption passwords by using and MD5 or SHA-1 hash functions. This blocks are placed at positions determined based on the passwords for each file.

Considering that we are dealing with integer values for the above information, the useful data in the header is 32B. The rest is padding, made up of random bytes. An attacker should have no way of distinguishing between the useful information inside the header and the padding.

3.2. The encrypted data

As seen in Fig.1, the archive should contain the two encrypted files. To increase the security of the solution the encrypted files will be split into blocks of fixed size and shuffled. Since a fixed shuffle scheme would have been trivial to undo, we used a key-based shuffle. The secret keys used for encryption are also used to guide the shuffle operations.

This approach also means that there is no need for any other secret keys, besides the two ones used for the encryption of the two files inside the archive.

4. STAGES FOR INFORMATION STORAGE AND RETRIEVAL

Fig. 3 shows the order of the operations which the user needs to perform during the data storage and retrieval phases. In the first row of the figure we have the stages needed to create the archive containing the

two encrypted files. On the bottom row we have the same stages in reverse order performed to retrieve the original data. Based on which of the two passwords the user decides to use, he will obtain the data for the first file or the second one, containing more sensitive data.

The actual compression step is simple but necessary in order to prevent the attacker from drawing any conclusions based on the size of the output file. Based on empirical evidence we decided to place the compression step at the beginning of the operation flow. This way the initial data has less entropy than the encrypted one and therefore the compression ratio is also slightly larger in most cases[7].

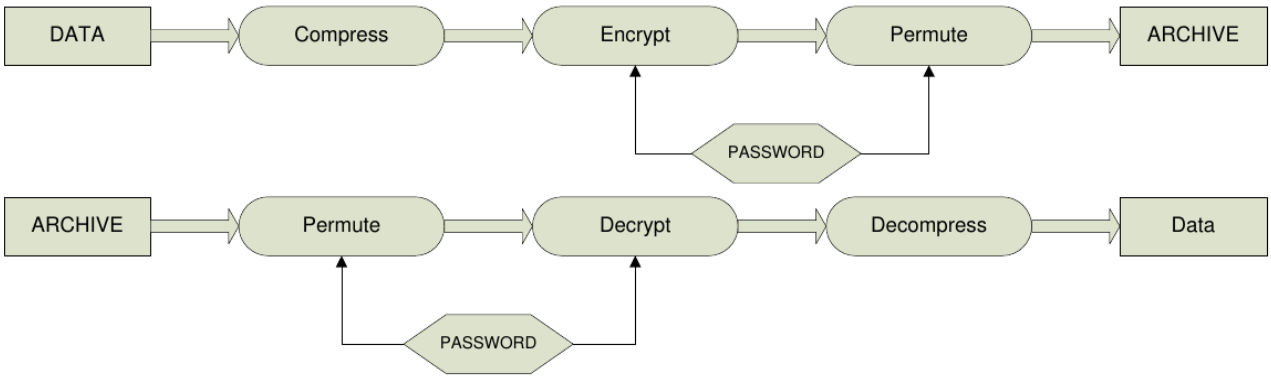


Fig. 3 – Operation flow for information storage and retrieval.

The actual compression step is simple but necessary in order to prevent the attacker from drawing any conclusions based on the size of the output file. Based on empirical evidence we decided to place the compression step at the beginning of the operation flow. This way the initial data has less entropy than the encrypted one and therefore the compression ratio is also slightly larger in most cases[7].

The encryption is also a very important step to ensure the security of the stored data. Since encrypted data is indistinguishable from sets of random bytes, this step also means that the attacker will have difficulty distinguishing between the encrypted files and the padding. However, this is also dependent on the ability of the random number generator to create strings with high entropy. Here the degree of security depends mostly on the algorithm used. We chose to use for our implementation the AES encryption algorithm, using a block size and key size of 128b, which is currently considered to be secure[6].

4.1. Key-based permutation

As an additional security measure, the entire data block is divided into blocks of the same size, which are then shuffled. The actual permutation is based on a key derived from the passwords used for the encryption and the number of blocks that the data has been divided into. The permutation we used is based on the algorithm presented in [8].

This permutation can be divided into three steps:

1) *Initialization of the permutation key.* Considering that the data we need to permute has been divided into N blocks we need to create a vector of the same size which will be the permutation key. Each element of the permutation key will be filled according to the ASCII value of a character from the password. Considering that the password which we use has a number of characters $n<N$ we are left with a number of $N-n$ positions in the key which we need to fill. This is done by adding two adjacent values and inserting the result on the first position. Afterwards, the values of the permutation key are brought into the interval [0,N). Considering that the character of the password has an ASCII value of X, the value of the element of the permuation key will be $X \bmod N$. This step ends when all the N positions of the permutation key have been filled.

2) *Elimination of duplicates.* If there are any duplicates inside the permutation key then the first value will be kept unchanged and the rest will be replaced by the value 0.

3) *Filling the blanks.* All the values of the permutation key must be brought into the interval *[1,N]*. All the values of 0 will be replaced with values which are not already present in the permutation key, starting from both ends of the key. This step ends when all the elements of the permutation key belong to the interval *[1,N]*.

An example for the creation of the permutation key, based on the password string "parola", for a permutation of 10 blocks, can be seen in Fig. 4.

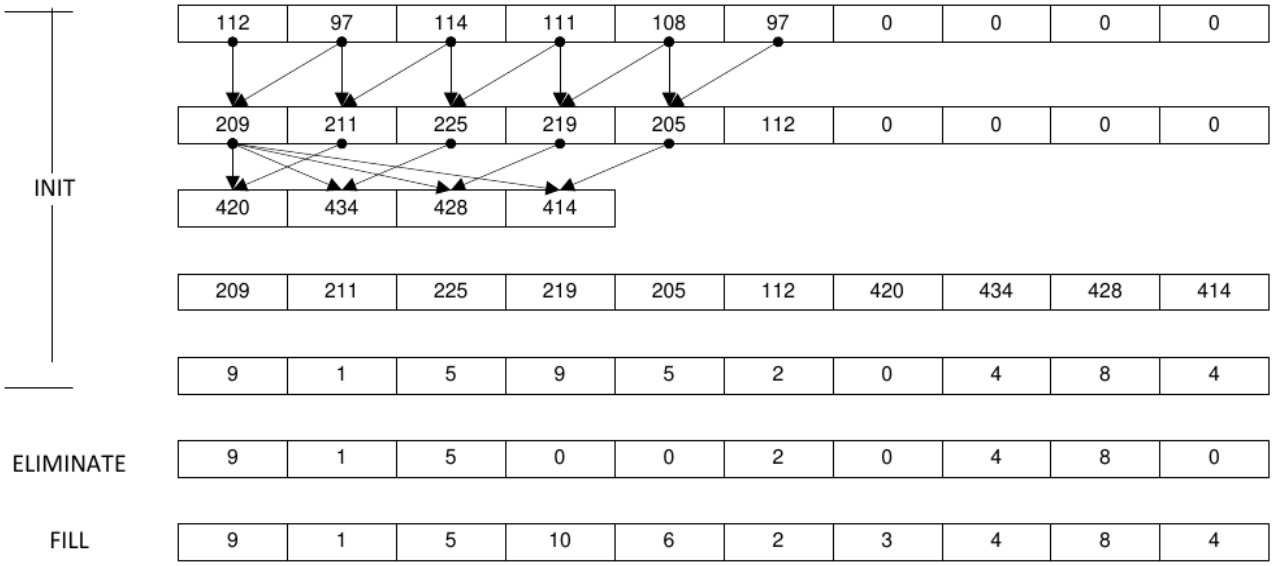


Fig. 4 – Example of the creation process of a permutation key.

The permutation is useful because it adds another level of protection for the data. An attacker would not be able to analyze the cyphertext until he has reversed the permutation. Since the permutation scheme is based on the passwords for the files it also does not require any other secret key from the user. After the end of the permutation stage, the result is the final form of the file containing the secret information. The process of retrieving the stored information is identical with the storage operations which we described in this section, but the necessary stages are executed in reverse order.

5. EXPERIMENTAL RESULTS

We tested the proposed solution using both text and binary files of different sizes. Our objectives were to validate the proposed solution and to evaluate it based on its performance and the size of the resulting archive. We were able to embed the two encrypted files of sizes between 100kB and 150MB into the archive, with padding between 10% and 30%, and also recover them successfully using the two passwords.

Regarding the performance of the proposed solution, we tested the required time for the each of the three main stages: the compression, encryption and permutation.

Fig. 5 and Fig. 6 show the dependency between the size of the files and the time needed for the data storage stages. Since there was a very large difference in the order of magnitude between the storage of very small files (1kB–100kB) and larger files, we chose to split this graph. Fig. 5 shows the performance for smaller files, upto 1MB, while Fig. 6 shows the same performance measurements, but for the experiments in which we used files between 1MB and 150MB.

We can see in these experiments the expected increase in processing time. However, considering that even for files of 150MB, we can see that each phase lasts less than 15 seconds, it is definitely slower than a simple compression or encryption. In total, for the largest experiment, the application would need 30 seconds to complete all the stages described in the previous sections. This should not be a problem for the user, since our solution is intended for the archival of sensitive data, not for data which is needed very often or data which is constantly updated.

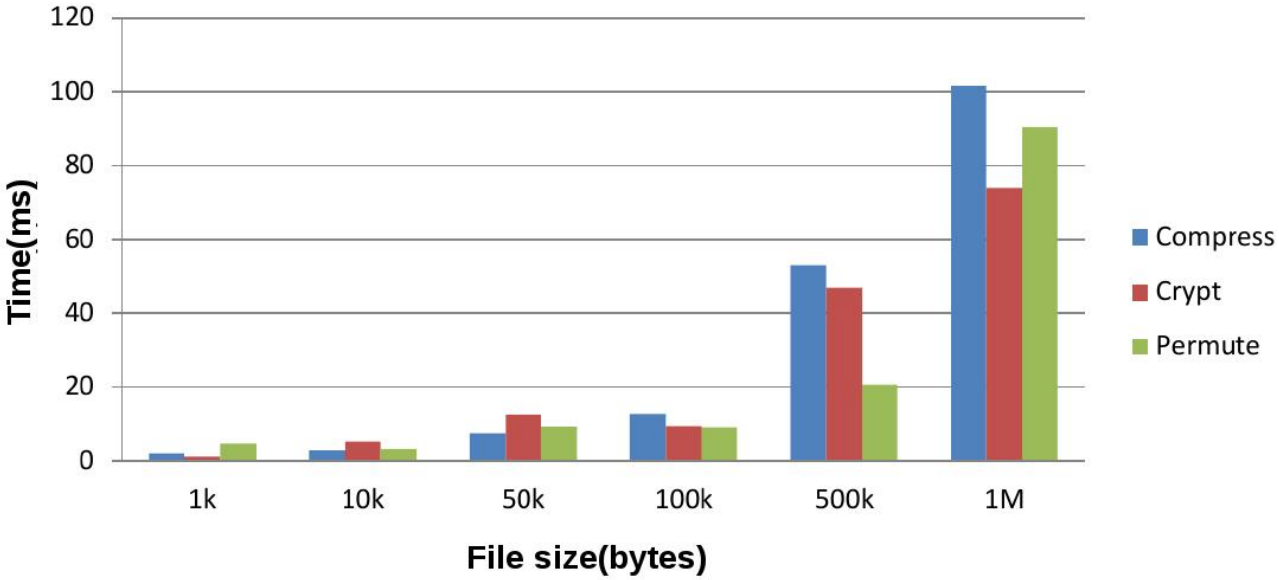


Fig. 5 – Performance of the storage operations for small files.

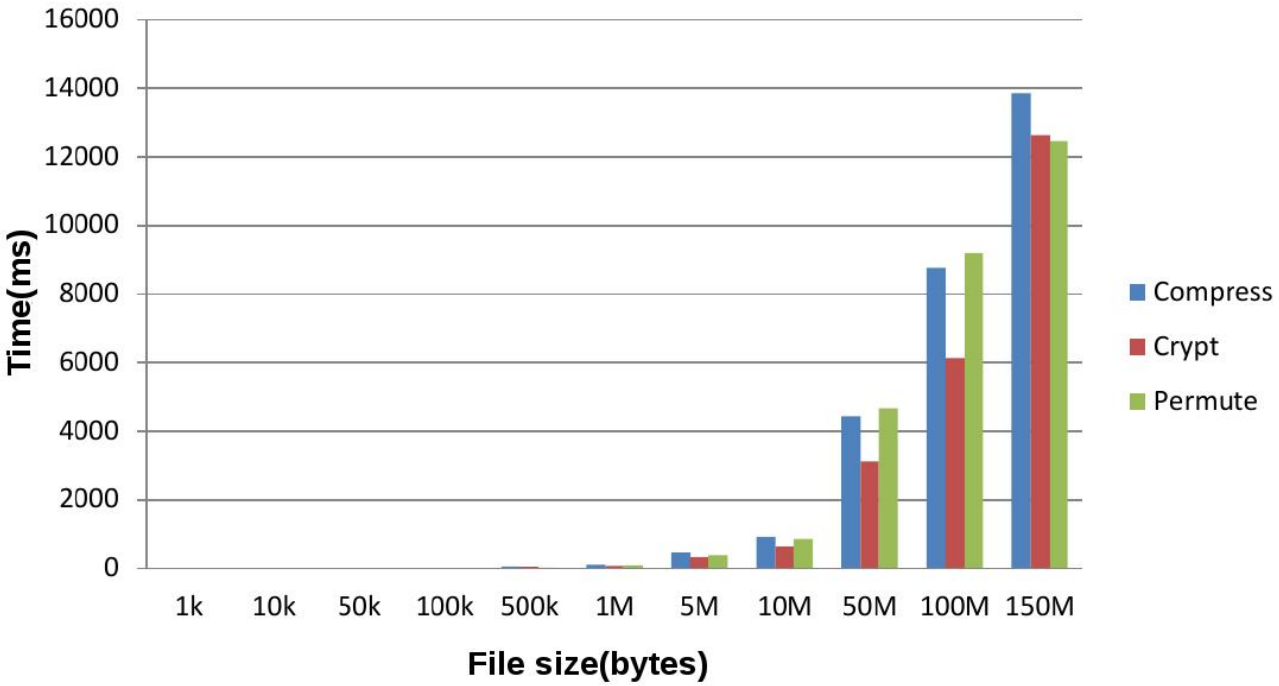


Fig. 6 – Performance of the storage operations for large files.

During our experiments we also found that the performance measurements were symmetrical. The measured performance was the same when we stored a set of files as when we retrieved them. This makes sense because they are basically the same operations, only executed in reverse order.

In Fig. 7 we represented the relative time needed for each stage of the proposed solution. It is evident that the permutation and the compression stages take the most time, while the encryption is highly variable, as opposed to the other stages.

Another important factor in our experiments was the size of the resulting archive. We needed to be sure that an attacker would not be able to discover the fact that there is any hidden data based on this information. We considered the following scenario. We inserted into the archive two files of different sizes, based on the algorithm described in the previous sections. In this case, the smaller file containing sensitive information is much easier to hide inside the archive, without raising any suspicions from an attacker. The difference in size could be attributed to the padding.

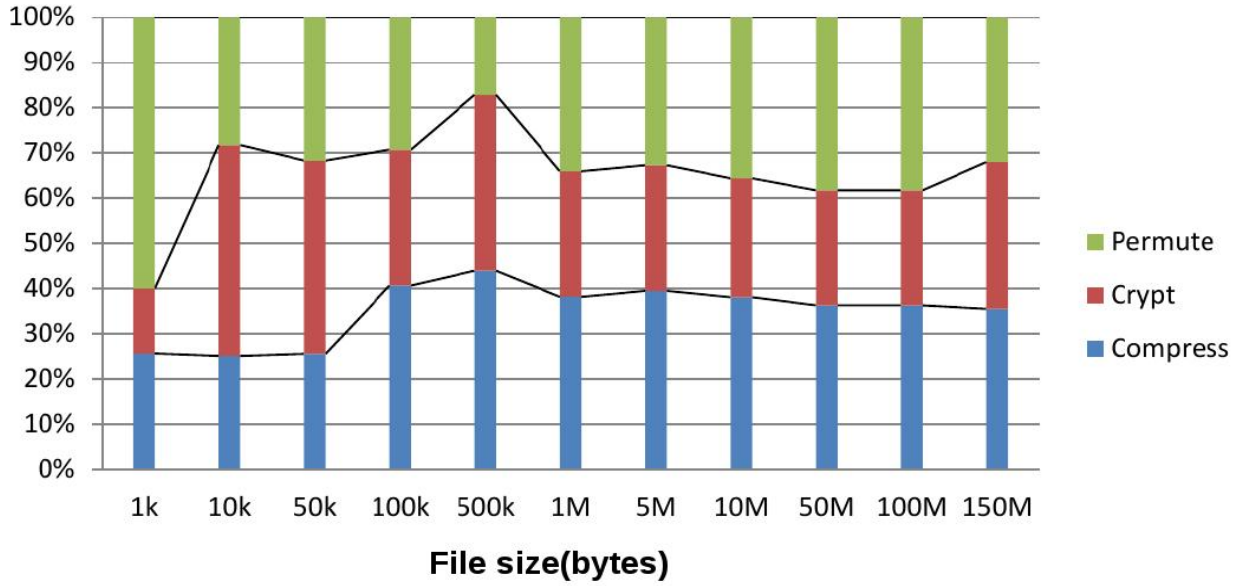


Fig. 7 – Relative performance for each stage.

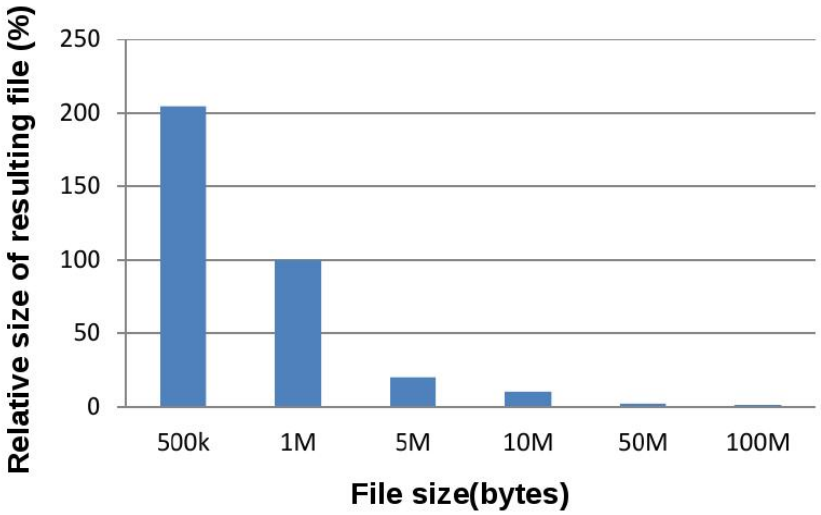


Fig. 8 – Relative increase in size for the storage of a 1MB file.

Fig. 8 shows the results of such a scenario. We considered a 1MB file which we needed to hide. The purpose of this set of experiments would be to find which would be the appropriate size of the second file inside the archive, so that the attacker would not be aware of the existence of the secret information. For this experiment, if we hide a 1MB file containing sensitive information, along with 5MB of data for the second file, we can see that the difference in size would be under 20%. We consider this to be the limit granting the user plausible deniability to the existence of the sensitive information. Any difference in the size of the archive below this limit could be considered to be a part of the padding.

6. CONCLUSIONS AND FUTURE WORK

In this paper we presented a solution to increase data confidentiality by creating an archive which contains an encrypted file, as well as a second hidden file, encrypted with a different key. The user is therefore granted plausible deniability regarding the existence of the second file. The security of the solution is directly dependent on the security of the encryption algorithm. For this solution we used the AES cypher which is currently considered secure.

We also added an additional level of security through a key-based permutation of the resulting archive after the encryption step. The role of this step is to further increase the difficulty for an attacker to analyze the archive and find the hidden data. Therefore, if the user is forced to reveal the password he can just decrypt the first file and an attacker would be unaware of the existence of any other useful data inside the encrypted archive. Any difference in size due to the existence of the second encrypted file could be blamed on the padding.

As future work we intend to further improve this solution. One direction would be to increase the number of tests and to see if we can increase the security of the hidden data, either by improvements for the storage and retrieval algorithms, or through the use of different methods for the random number generator, permutation or cryptographic algorithms. Another direction for future work could be the investigation of the actual storage methods for such confidential data. Since Cloud storage services are becoming widespread, there is a need to improve the security of such data and to make sure that the users can trust that their data is not accessed by any unauthorized entitites, even if they do not have direct control over the storage environment. Ensuring data confidentiality is a very challenging subject, which has the potential to improve many security systems, as well as the usability of existing storage solutions.

REFERENCES

1. Anderson, Ross, Roger Needham, and Adi Shamir, *The steganographic file system*, Information Hiding. Springer Berlin Heidelberg, 1998.
2. A. Czeskis, D. Hilaire, K. Koscher, S. Gribble, T. Kohno, B. Schneier, *Defeating Encrypted and Deniable File Systems*, HOTSEC'08, 2008.
3. Czeskis, Alexei, *et al.*, *Defeating encrypted and deniable file systems: TrueCrypt v5. 1a and the case of the tattling OS and applications*, 3rd USENIX HotSec (2008).
4. McDonald, Andrew D., and Markus G. Kuhn, Stegfs, *A steganographic file system for linux*, Information Hiding, Springer Berlin Heidelberg, 2000.
5. Canetti, Rein, *et al.*, *Deniable encryption*, Advances in Cryptology—CRYPTO'97, Springer Berlin Heidelberg, 1997. 90–104.
6. Akkar, Mehdi-Laurent, and Christophe Giraud, *An implementation of DES and AES, secure against some attacks*, Cryptographic Hardware and Embedded Systems–CHES 2001. Springer Berlin Heidelberg, 2001.
7. Balakrishnan, Kedarnath J., and Nur A. Touba, *Relating entropy theory to test data compression*, Proceedings IEEE European Test Symposium. 2004.
8. S.M. Hussain, N.M. Ajlouni, *Key Based Random Permutation*, Journal of Computer Science 2 (5): 419–421, 2006

Received July 15, 2013

THE PUBLISHING HOUSE
OF THE ROMANIAN ACADEMY

PROCEEDINGS OF THE ROMANIAN ACADEMY, Series A,
Volume 14, Special Issue 2013, pp. 365–372

APPLICATIONS OF NATURAL COMPUTING IN CRYPTOLOGY: NLFSR BASED ON HYBRID CELLULAR AUTOMATA WITH 5-CELL NEIGHBORHOOD

Radu DOGARU[*], Ioana DOGARU[*]

[*] University "Politehnica" of Bucharest, Dept. of Applied Electronics and Information Engineering, Natural Computing Laboratory
Corresponding author: Radu DOGARU, E-mail: radu_d@ieee.org

This work reviews some important issues in natural computing that are of interest for cryptology. In particular we focus on our recent results in defining a particular class of hybrid cellular automata (HCA) with 5 cells neighbourhood as discrete-time chaotic maps, with very good cryptographic properties. Such structures are better alternatives to other chaotic maps since they are using the hardware resources with a maximal efficiency and have no transient times associated to the convergence to the main cycle. Based on the algebraic normal form (ANF) representation of the HCA-rule it is proved that our HCA are in fact nonlinear feedback shift registers NLFSR, recently gaining increasingly interest for cryptographic applications (such as stream ciphers). It is shown that the FPGA resources are optimally allocated as a consequence of using the ANF representation of cells.

Key words: natural computing, cellular automata, chaotic dynamics, nonlinear feedback shift register, pseudo-random number generators, algebraic normal form

1. INTRODUCTION

Cryptology applications may be successfully approached by applying certain natural computing [1] techniques. Particularly, in this work we focus on a particular class of nonlinear dynamic systems inspired from nature; namely, the cellular automata. Cellular automata fit in the more general network model represented in Fig.1. At its very fundamental level (mathematical description) the network is a nonlinear dynamic system. Cells are associated with state variables (scalars or grouped in a vector structure) while connectivity links represent (nonlinear) functional relationships between states. These functional relationships include a certain number of parameters (the cell's gene in Chua's parlance [2], depicted as w_{ij} on links in Fig.1). Finding these parameters (as well as the functional expressions) such that the model fits specific goals represents the design tasks. For instance, two goals may be of interest in cryptology:

i) Designing a network capable to generate pseudo-random sequences that are difficult to decode in case of attacks (in this case genes are tuned such that in the end the network respond properly to certain batteries of tests [3]. Often the nonlinear network is designed such that it generates chaotic sequences [4][5][6] but more traditional models such as linear and nonlinear feedback shift register (NLFSR) [7][8] and cellular automata CA [9] also fit in the same category;

ii) Using partially available information from encrypted messages (as an intruder) and consider the nonlinear network as an adaptive one (e.g. using neural network paradigms) that is eventually capable to "decrypt" the structure behind a pseudo-random generator or a more sophisticated cryptographic system. This second goal is grounded on Takens's embedding theorem [10] and further methods built upon it (for instance [11]). Takens theorem allows the creation of hidden state variables in a network model based on a limited (but large enough) set of past observations. Since the accuracy of the model is better for lower number of state variables in order to avoid decryption based on Takens theorem, an extremely large state space for the cryptographic sequence generator must be considered. From this perspective the low (1-dimensional) logistic map (with no delay between transmitted samples, or other scrambling methods) is very fragile against attacks using Takens theorem [4]. On the other hand cellular automata CA were recognized as very good dynamic networks to generate cryptographic sequences since the number of their cells can be chosen as an arbitrarily large one.

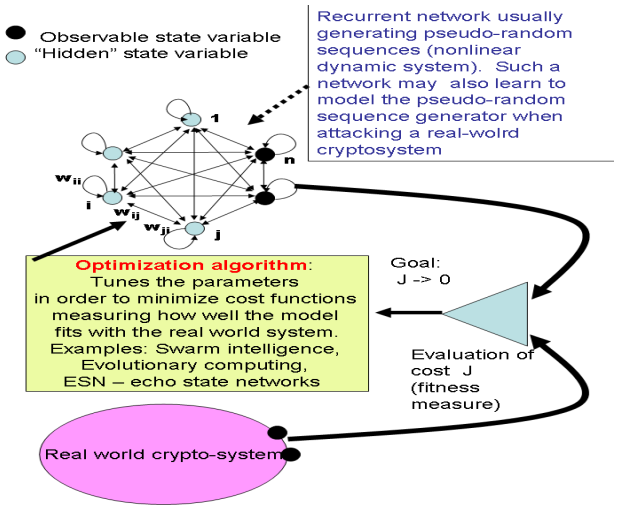


Fig. 1 – A natural computing approach to problems in cryptology: a) generating good pseudo-random sequences to be exploited in cryptosystems; b) to identify and replicate cryptographic systems (attack problems) based on consecutive observables from an encrypted transmission (real-world crypto-system).

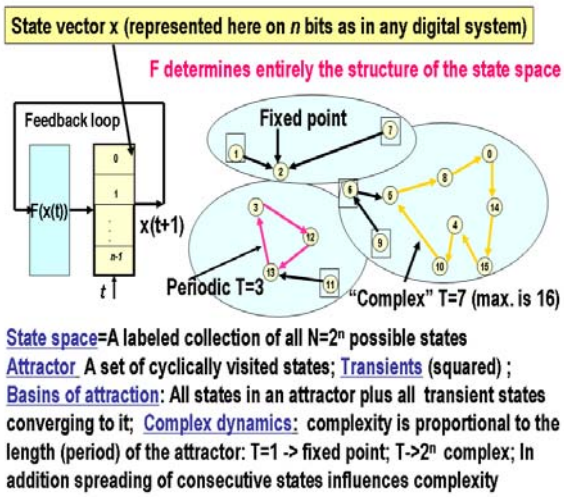


Fig. 2 – General structure of a nonlinear map in a digital implementation (discrete-time, finite precision) and the main concepts associated with it.

Mathematically, the nonlinear dynamic network is an ordinary differential equation (ODE) in the case of continuous time or a finite difference equation in the case of discrete time. In the next, we will focus only on discrete-time systems associated with their digital system implementations, ensuring reliable reproduction of identical networks as needed often at both transmitter and receiver.

In this paper, recent results on identifying a large class of cellular automata with good cryptographic properties are exposed. Such systems may be further used in various cryptosystems as pseudo-random number generators. First, in section 2, a general model for the nonlinear dynamical system is presented using discrete-time and finite precision representation of state variables. Several performance descriptors are reviewed and introduced. Particularly we will stress on conservative systems (i.e. without transients) with maximal length cycle and good randomness (an equivalent of chaotic behaviour for the case of finite state space and discrete time). Section 3 presents basic notions of cellular automata, introducing the hybrid cellular automata (HCA) with 3 and 5 cells neighbourhoods. In Section 4 a methodology is given for representing local rules in algebraic normal form (ANF) and for the identification of all nonlinear HCA (HCA NLFSR) with very good cryptology properties. Properties and conclusions are also included here.

2. DISCRETE-TIME NONLINEAR DYNAMICAL SYSTEMS AS CHAOTIC MAPS, PERFORMANCE DESCRIPTORS

In the general case, a discrete-time nonlinear dynamic system is given by equation: $\mathbf{x}(t+1)=F(\mathbf{x}(t))$, where $\mathbf{x}$ is a *state variable.* The above is called a *map* and if the dynamic behaviour is chaotic the map is said to be *chaotic.* In recent year chaotic maps gained increased interest for applications in cryptology [5][6]. Quite often they are used to build cryptosystems where the underlying dynamical system is used as pseudo-random number generator. A widely known example of a chaotic map is the one dimensional logistic map [12][13] where $F=\lambda x(1-x)$. From a practical perspective (digital implementation, where all state variables can be associated with a unique register with n bits) any nonlinear map can be associated with an automaton-type as presented in Fig. 2. For instance, if the unique x state variable of the logistic map is implemented in a 32-bits floating point representation the associated register in Fig. 2 has $n=32$ bits and the nonlinear function $\mathbf{F}$ is implemented as a combinational digital circuitry. Often, in digital implementations fixed-point representations of the state variables are preferred since they ensure a better efficiency (less occupied

resources). In [14] an interesting comparative study of various chaotic maps using various sizes *n* with fixed or floating point implementations in FPGA is given. In [15] we show that the general structure of a digital implementation for a nonlinear map in Fig.2 includes cellular automata (CA) as particular case. Moreover, the CA have several advantages over nonlinear maps, such as the lack of transients (for certain kind of cellular automata called *conservative* CA), easiness of scaling to high dimensionality *n* (with direct implications for decreasing the probability of successful attack), better efficiency in hardware (*e.g.* FPGA), to name just a few. More details about cellular automata as chaos generators and their design are given in Section 3. In the following we introduce the main performance descriptors and discuss them from the cryptology perspective. As a general rule, in cryptology we are interested to design systems with very long cycle length *L* (as close as possible to the maximal length i.e. $N = 2^n$), large state dimension *n*, and a "chaotic" character of the longest cycle, no transients, and an efficient and scalable VLSI implementation:

Transients: They are also called "ephemeral states" (Fig.2) since they are never visited twice during the dynamics of the system. Most chaotic maps reported so far have such transients and, to our knowledge there is no consistent theory to predict the transient length for a given system (although some useful computational evaluations were done in [13]). It is also clear that the initial state influences the transient length (i.e. the number of iterations until entering a *cycle*). For cryptographic application is useful to avoid the existence of transients since they are not useful states and consequently their presence will diminish the "chaotic" cycle length ($L<N$) also requiring some lost cycles until entering the useful cycle. Recently [16] within the cellular automata framework, *conservative* CA was defined (it also apply to any network model in Fig.2) as those having the property that *no state is a transient or ephemeral state*. Both LFSR and NLFSR (linear feedback shift register) used for long as pseudo-random sequence generators have this property as well. The cellular automata in this paper are also conservative and consequently they have no transients.

Cycle length (L): As seen in Fig.2 for different feedback functions **F** (parameters of **F** often represent the encryption key) different state space profiles are obtained. They consist of cycles of various lengths. Ideally, for cryptographic applications, the state space profile should be represented by a single cycle with the maximal length $L = N = 2^n$. In practice solutions where one very large cycle with $L \cong N$ coexists with several very small cycles are acceptable. The closest is L/N to 1 the better. For practical reasons it is also convenient that a particular state (such as "all bits in 0") is enclosed in the dominant cycle (the one with maximal length).

Randomness (degree of chaos): A very long cycle is not necessarily a random one. A good counter-example is the counting automaton (known as "counter" in digital circuit parlance). It has a maximal cycle length $L=N$ but the transition from one state to the consecutive one is rather smooth, often only one bit is changing. Moreover, altering one bit in a counting state gives no dramatic influence on the trajectory. According to the classical chaos theory, a small change in the state variable must influence dramatically the values of the consecutive states (many different bits in our case). Since here we discuss about finite state space, the use of Lyapunov exponents (defined in the context of continuous state variables) is not suitable. Instead, in [17] we introduced a randomness measure that may be conveniently computed. It characterizes very well the randomness of any cycle. We are in particular interested by the randomness of the *dominant cycle*. The measure of randomness was defined observing that in a "chaotic" automata the average Hamming distance between consecutive binary vector states (as given by the *n* cell outputs) becomes *n/2* instead of 1 for counters. Therefore, for any arbitrary cycle C_j of length L_j a *scattering coefficient* S_j is defined by averaging the Hamming distances between all consecutive binary vector states in that cycle:

$$S_j = \frac{1}{nL_j} \sum_{k=1}^{L_j} \sum_{i=1}^{n} \left| x_i(k) - x_i(k-1) \right| \tag{1}$$

where *k* is the time index of consecutive states in the cycle *j*. A *degree of chaos* λ_j is defined such that it becomes maximum if $S_j = 0.5$ and zero for the extreme, non-chaotic cases of both fixed points and period 2 cycles (with $S_j = 0$ and $S_j = 1$ respectively):

$$\lambda_j = 1 - \left| 2S_j - 1 \right| \tag{2}$$

The *degree of chaos* may be regarded as qualitatively similar to the Lyapunov exponent used in continuous-state systems to characterize chaotic behaviours. In our case its largest value is $\lambda_j = 1$ indicates the highest degree of randomness in a finite-length cycle of an automata network.

Efficiency of hardware utilization: Such a descriptor indicates: i) how many basic devices are needed to implement the feedback function F for a given *n* and ii) how well the state space is used (*i.e.* the ratio *L/N*). As shown in [15] usual chaotic maps (logistic in that case) would need far more hardware resources than hybrid cellular automata introduced first in [18]. For instance, according to [15] in the case *n=32* , 1317 LUTs (basic logic cells in FPGA) are needed while only 32 LUTs (in general *n* LUTs) are needed for the HCA101 cellular automata. On the other hand, the HCA101 has a far better utilization of the state space, for example, as shown in [15] in the case *n=21*, $L = 2097151 = 2^{21} - 1 = N - 1$ for HCA with ID=101 *i.e.* all states except 1 are included in the dominant chaotic cycle (also there are no transients) while for the same size *n* of the register the logistic map with $\lambda = 3.7$ (proved to be chaotic in floating point implementations) has the length of the dominant cycle only $L = 883$ (*i.e.* a fraction of only 0.0004 of all states), all other states belonging to transients! Such results indicate why it is desirable to consider a special case of cellular automata, the hybrid cellular automata (HCA) instead of (classical) chaotic maps. They are conservative maps and have the highest degree of randomness dominant cycle.

3. HYBRID CELLULAR AUTOMATA

Cellular automata have a long history of their use in cryptology. Many patents dealing with cellular-automata based cryptographic systems are issued, (first patent on this issue [19]) and their study is the focus of many research papers (*e.g.* [9][20][21]). The cellular automata model fits the general model of a nonlinear map implemented as a digital system, each bit of the register being now associated to one cell and the feedback function **F** is now defined as a collection of *n* similar local (Boolean) functions acting in a *neighbourhood* of *m* cells. The reason why cellular automata are so popular in cryptology is the easiness to scale them to arbitrarily large *n* (as discussed above, a large size of the state space makes cryptographic attacks more difficult) due to the locality of the **F** mapping. Among the first cellular automata types that were widely investigated are the elementary cellular automata (ECA) [Wolfram 1983] where the neighbourhood size is *m=3*. Randomness was then associated with the evolution of cellular automata with rule (ID) 30, also adopted as random number generator in Wolfram's "Mathematica". Consequently, Chua [16] did an exhaustive research on all possible 256 ECA and introduced the concept of *conservative* cellular automata. As shown, the ECA with ID=30 is not a conservative cellular automata, consequently it has the transients. In [16] it is shown that CA with odd number *n* of cells governed by rules ID = 45 (and its other 3 equivalents ID = 75,89, and 101) and the complemented outputs ID = 154 (or its equivalents, ID = 166, 180, 210) are the only *non-linear rules* leading to *conservative* dynamics with a high degree of complexity. Independently, in [17] we were able to show that in addition to this property, the cellular automata with ID = 45,75,89,101 have very good randomness properties, although the lengths of the dominant cycle are not maximal for any *n*. In addition these automata networks allow synchronization between a transmitter and a receiver with identical structure using only 1 bit (binary synchronization), a feature that may be conveniently exploited particularly in communication systems. The good randomness and cryptographic properties of CA with ID=101 and it equivalents is confirmed in [9] using standard batteries of statistical tests [3]. Later, in [15] we introduced the concept of a hybrid cellular automata (HCA) demonstrating that for a proper choice of a mask vector of size *n* (complementing the outputs of some ID = 101 cells) one can maximize *L* such that it will reach a value very close to *N* (thus, improving the cryptographic properties of the random number generator [30]). Several applications were proposed [22][23], particularly interesting is the one in [23] where such a HCA is used as "chaotic counter" to scan a video sensor leading to a very compact yet versatile system which ensures compression, encryption and spectrum distribution at the same time and with very little hardware requirements. Such solutions are very convenient for low power remote sensing applications (*e.g.* surveillance etc.). As seen, instead of various parameters of the chaotic maps (*e.g.* the λ parameter) in the case of CA or HCA maps the rule (also called ID, to be detailed next) takes the role of parameter, being associated with part of the key space. The masks in the HCA may represent part of the encryption key as well as the initial state. In order to increase the key space one needs to consider larger neighbourhoods such

that more IDs will be identified as useful in terms of good cryptographic properties. Consequently, in Section 4 we discuss the case *m=5* (5 cells neighbourhood) and locate many other ID (from a huge space of 2^{32} – about 4 billion) to generate HCA with good cryptographic properties. Consequently the key space is considerably expanded than in the case of *m=3* neighbourhood.

3.1. HCA Structure

To explain the HCA structure we consider first the case *m*=3 (3 cells neighbourhood). It expands naturally to a 5-cell neighbourhood. Fig. 3 presents two *m=3* HCA automata structures that were successfully tested for having both good cryptographic (as exposed in section 2) and binary synchronization properties. Note that they can be operated in either autonomous mode (as is the case in the transmitter system Tx) or with one input forced by the synchronization signal (as is the case in the receiving system Rx).

The discrete-time dynamics of the hybrid cellular automata (HCA) in Fig. 3 is given by the next equation, which applies synchronously to all *n* cells (a cell is identified by an index $i \in \{1,2,..n\}$):

$$x_i^T(t+1) = m_i \oplus Cell\left(x_{i-1}^T(t), x_i^T(t), x_{i+1}^T(t)), ID\right) \tag{3}$$

where the upper index "*T*" stands for the transmitting CA counter, $\oplus$ is the logical XOR operator and $Cell(u1,u2,u3,ID)$ is a Boolean function with 3 binary inputs (*u1,u2, and u3*), also called the CA (local) rule. A periodic boundary condition is also assumed *i.e.* the leftmost cell (*i=1*) is connected to the rightmost one (*i=n*). The binary *mask vector* $\mathbf{m} = [m_1, m_2, .., m_n]$ can be optimized [18] (so far our programs and limited computational time allow up to $n \le 29$) to obtain a maximal cycle length ($r = L/2^n \to 1$). Since there are many near-optimal length masks, the mask itself can be considered as part of the encryption key. There are many possibilities to increase the key space. The one described in detail in the next section assumes a larger neighbourhood and consequently many types of cells ensuring the desired properties. One possibility, recently investigated, is to consider an alteration of the regular cellular topology into a "small worlds" one, as shown in Fig. 3 (right side). Essentially the model is described by the same equation (3) where the mask can be removed (*i.e.* all $m_i = 0$) but where the optimization of the maximal cycle length may now be achieved using a random search process in a space of permutations (one or more pairs of outputs are swapped as shown in Fig.3). The mask is now replaced by the set of swapping pairs (i_{sw}, j_{sw}). The above equation (3) extends easily to larger neighbourhoods such as *m*=5 considered herein by adding 2 additional inputs to the cell, located on the rightmost and leftmost positions (*i-2,* and *i+2*). For any neighbourhood the relationship between inputs and the output local CA rule can be characterized in two different ways and conversion functions are available via [24]: a) ***Truth-Table*** *(TT)* representation: This is the most widely used representation. The rule is characterized by a binary vector $Y = [y_{N-1}, y_{N-2}, ..., y_0]$. Its representation in decimal basis is called a rule identifier (ID). The output y_k is a binary number assigned to the cell's output when its inputs ordered as a binary vector $[u_n, u_{n-1}, .., u_1]$ are the binary representation of *k;* b) ***Algebraic Normal Form (ANF):*** This form is described by a binary vector $C = [c_0, c_1, ...c_N]$ (using the method in [24] a unique conversion from **Y** to **C** and vice-versa exists) such that its coefficients are multipliers of an algebraic representation on the GF_2 exemplified next for the case of *m=3* neighborhood:

$$y = c_0 \oplus c_1u_1 \oplus c_2u_2 \oplus c_3u_2u_1k_3u_3 \oplus c_4u_3 \oplus c_5u_3u_1 \oplus c_6u_3u_2 \oplus c_7u_3u_2u_1 \tag{4}$$

Note that in general (for any size *m* of the neighbourhood) c_k is the multiplier of a product (logical AND) of all input variables in a binary vector $[u_n, u_{n-1}, .., u_1]$ corresponding to 1 in the associated binary vector representing *k.* For example, in the case of *m=3* (above equation) for $k = 5 = 101_2$ only the inputs corresponding to 1 in the input string $u_3u_2u_1$ are selected to be multiplied resulting in the term $c_5u_3u_1$.

The ANF representation is extremely useful for FPGA implementations since it allows a direct translation into a simple VHDL line describing the entire HCA structure [25]. The resulting synthesized structure has a very efficient use of the resources, better than reported in other works for similar automata [26]. On the other hand, ANF is very useful to reveal whether the automata network is a linear one (*e.g.* in the same category with LFSR and other hybrid cellular automata that are mostly reported so far, typically

with *m=3* and rules ID=90, ID=150 [27]) or a nonlinear one (from the same category with NLFSR, recently gaining a lot of interest [7][8][28] as having a better immunity to attacks than LFSR). A nonlinear automaton has at least one term with more than 2 inputs in the ANF equation (4).

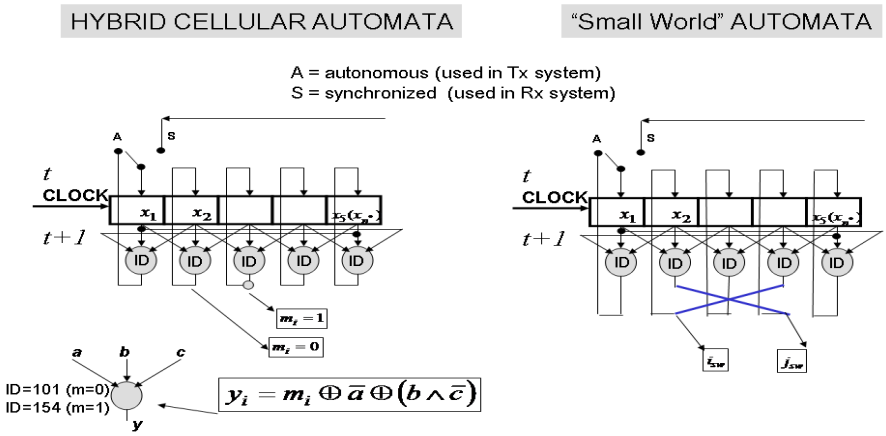


Fig. 3 – Two structures of HCA with *m=3* neighbourhood: a) left – the "standard" HCA, b) the "small-worlds" HCA where some of the outputs are swapped. Such chaotic counters may be operated in either autonomous or synchronized mode.

4. RESULTS AND CONCLUSIONS

In order to investigate the very large family of 5-cells neighbourhood automata (there are $2^{2^5} = 4294967296$ different cells possible) we adopted the following strategy:

i) a set of software tools was developed as follows: a) Matlab tools for conversions between ANF and TT representations; b) a Matlab tool to establish the attractor profile revealing the cycles, their lengths and their associated randomness for an arbitrary number of cells *n* (Fig. 4). This program is used during a search process to locate ID-s with potential for good cryptographic properties [3][30]; c) A compiled C program to optimize the mask in order to get a very long cycle (state "0" corresponding to all bits equal to 0 is included in the longest cycle) and evaluate the average number of cycles for binary synchronization (Figure 5). Examples are given for ID=1347465135 corresponding to the ANF: $y = 1 \oplus u_5 \oplus u_1 \oplus u_5 u_1$.

Note than finding the optimal mask is a computationally intensive processes involving the running of long cycles for several thousands of trials. The computational complexity is $O(2^n)$ and it takes about 3 minutes for *n*=17 and 4 times more for *n*=19.

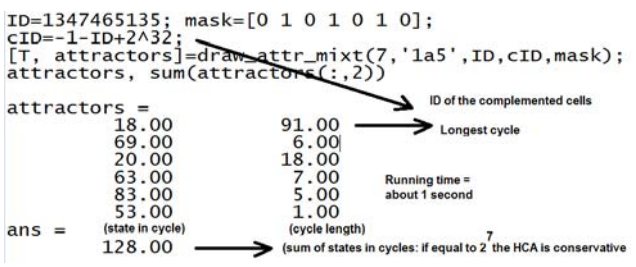


Fig. 4 – Detecting the attractor structures for a given ID and mask.

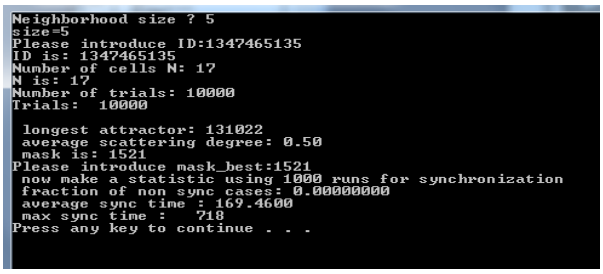


Fig. 5 – Optimization of mask.

ii) We considered the ANF representation to describe all possible families of 256 HCA corresponding to 3 inputs in the 5-cell neighbourhoods and use the conversion software to calculate ID and then evaluate the profile of attractors keeping only the desired CA. The situations that are symmetrical with respect to the central cell were not considered because they can be easily recovered by simply replacing $u_k = u_{6-k}$. A synthesis of the results is given in the following table.

Distribution of inputs in the neighbourhood	ID	ANF
[u5,-,u3,-,u1] 8 HCAs with good cryptographic properties	2947502160 2779097770 4126476810 2863290970 + 4 complemented versions of the above	$y = 0 \oplus u_3 \oplus u_1 \oplus u_5u_1$ $y = 0 \oplus u_5 \oplus u_1 \oplus u_3u_1$ $y = 0 \oplus u_5 \oplus u_1 \oplus u_5u_3$ $y = 0 \oplus u_5 \oplus u_3 \oplus u_3u_1$
[u5,u4,-,u2,-] and [u5,-,u3,u2,-]	NONE	-
[-,-,u3,u2,u1]	4043247360 4027576560 4279173360 4042264560 + 4 complemented versions of the above	$y = 0 \oplus u_2 \oplus u_1 \oplus u_3u_2$ $y = 0 \oplus u_3 \oplus u_1 \oplus u_2u_1$ $y = 0 \oplus u_3 \oplus u_1 \oplus u_3u_2$ $y = 0 \oplus u_3 \oplus u_2 \oplus u_2u_1$
[-,u4,u3,u2,-] equivalent to [u3,u2,u1] (3 cells neighborhood)	8 Boolean functions already described in [Dogaru HCA]	The above formulare where u_k is replaced with u_{k+1}

At a closer inspection of the ANF formulae in the above table a very interesting conclusion follows: In all cases the same logical functions with 4 inputs is performed namely: $y = m_i \oplus u_a \oplus u_b \oplus u_cu_a$ where m_i is bit *i* of the mask and all 6 possible permutations of the indices *a,b,c* are possible. Further studies indicates that the relationship between *a,b* and *c* leading to good cryptographic HCA is: $a - b = b - c = h$ where *h* is an integer. For instance, if $h = 2$ $(a,b,c) = (5,3,1)$ and all 6 permutations leading to the IDs from the first line of the above table. Higher values of *h* will make sense for neighbourhoods larger than 5. For instance, h = 3 will lead to $(a,b,c) = (7,4,1)$ corresponding to *m=7* cell neighbourhood. For any of these possible combinations one may optimize masks as seen in Fig. 5. For instance ID=1347465135 has the best mask found so far 19801 leading to *L=N-1=131071.* In this case as for many other masks of the large family of CA discussed above, the randomness coefficient is maximal, *i.e.* $\lambda = 1$. A detailed analysis using standard batteries of statistical tests reveals that all HCA from the family generated by the unique logical functions with 4 variables have very good cryptographic properties [29] expanding similar results reported by other authors [9] for the elementary CA with ID=101 which also uses a form of the above logical function as local rule.

Concluding, in this work we approach the design of good random number generators using natural computing concepts, mainly the one of cellular automata. It is shown that all major paradigms for cryptographic generators (chaos based, linear and nonlinear shift register and cellular automata) fit in the same unique model of a nonlinear network with a binary state vector represented on *n* bits. It is shown that comparing with chaotic maps the use of hybrid cellular automata as nonlinear maps brings the advantage of an efficient use of the state space (no transients). Moreover, the case of 5-cell neighbourhood (using cells with 4 inputs, such that each HCA cell will correspond to 1 basic computational element (LUT or LE) in FPGA technology as already was demonstrated in [25]. The algebraic normal form was conveniently used for synthesis since equation (4) translates directly into one VHDL line ensuring the description of the underlying CA. Further research will focus on identifying all cryptographically useful HCA-NLFSR with 4 and 5 inputs rule in a 5-cell neighbourhood. For such cells, a VHDL description is already available which shows after synthesis that 2 LUTs are needed for each HCA cell (doubling the necessary hardware resources for the same *n* when compared to the case of 3-inputs from the 5-cell neighbourhood). Preliminary results show that among cells with 4-inputs, the HCA with ID=3432828060 was found as having similar cryptographic properties with the HCA family discussed above, but in addition it has this properties for an arbitrary *n* number of cells (the other HCAs do not have the "no transients" property for even *n.* Such a property is particularly useful in applications such as [23] in addressing square image sensor that will require an even number *n* of bits. Another further open question is to develop a systematic theory for finding the optimal masks analytically and to explain why only a very small number (among the many possible IDs) are both conservative and also have good randomness properties. Such problems are recognized as open problems in the NLFSR research community as well [29].

REFERENCES

1. Leandro N. De Castro, *Fundamentals of Natural Computing: Basic Concepts, Algorithms, and Applications*, Chapman & Hall/CRC Computer, 2006.
2. L. O. Chua, *CNN: a Vision of Complexity*, International Journal of Bifurcation and Chaos, Vol.7, No.10, pp. 2219–2425, 1997.
3. National Institute of Standards and Technology, Federal Information Processing Standards Publication 140-2: *Security Requirements for Cryptographic Modules*, US Government Printing Office, Washington, 1999.
4. G. Alvarez, S. Li, *Some basic cryptographic requirements for chaos-based cryptosystems*, Int. J. Bifurcation Chaos Appl. Sci. Eng. 16, pp. 2129–2151, (2006).
5. N.K. Pareek, Vinod Patidar, K.K. Sud, *Cryptography using multiple one di-mensional chaotic maps*, Commun. Nonlinear Sci. Numer. Simul. 10 (7) (2005), pp. 715–723.
6. W. M. Tam, F. C. M. Lau, and C. K. Tse, Digital Communications With Chaos. Oxford, U.K.: Elsevier, 2007.
7. Tomasz Rachwalik, Janusz Szmidt, Robert Wicik, and Janusz Zablocki, *A Generation of Nonlinear Feedback Shift Registers with special-purpose hardware*, Cryptology ePrint Archive, Report 2012/314, June 2012, http://eprint.iacr.org/2012/314
8. E. Dubrova, *How to speed-up your NLFSR-based stream cipher*, in Proceedings of Design, Automation & Test in Europe Conference & Exhibition (DATE '09), pp. 878–881, 2009.
9. Franciszek Seredynski, Pascal Bouvry, Albert Y. Zomaya, Cellular automata computations and secret key cryptography. Parallel Computing 30(5-6): 753–766 (2004)
10. F. Takens, *Detecting strange attractors in turbulence*, in D. A. Rand and L.-S. Young. Dynamical Systems and Turbulence, Lecture Notes in Mathematics, vol. 898. Springer-Verlag. pp. 366–381, 1981.
11. M.B. Kennel, H.D.I. Abarbanel, *False neighbors and false strands: A reliable minimum embedding dimension algorithm*, in Phys. Rev. E, VOl. 66, 026209 (2002) [18 pages].
12. A. Kanso, N. Smaoui, *Logistic chaotic maps for binary numbers generations*, in Chaos, Solitons and Fractals, 40 (2009), pp. 2557–2568.
13. Adriana Vlad, A. Luca and M. Frunzete, *Computational Measurements of the Transient Time and of the Sampling Distance That Enables Statistical Inde-pendence in the Logistic Map*, Lecture Notes in Computer Science, 2009, Volume 5593/2009, 703–718.
14. P. Giard, G. Kaddoum, F. Gagnon, C. Thibeault, *FPGA implementation and evaluation of discrete-time chaotic generators circuits*, In proceeding of: IECON 2012 - 38th Annual Conference on IEEE Industrial Electronics Society, pp.3221–3224, 2012
15. R. Dogaru, *HCA101: A chaotic map based on cellular automata with binary synchronization properties*, in Proceedings of The 8'th Int'l Conference on Communications-COMM2010, 10-12 June, Bucharest, Romania, pp. 41–44.
16. L.O. Chua, A Nonlinear Dynamics Perspective of Wolfram's New Kind of Science (Vol I-IV), World Scientific Series on Nonlinear Science, Series A - Vol. 57, 68, 76, World Scientic Publishing Company, 2006, 2009, 2011.
17. R. Dogaru, I. Dogaru, and H. Kim, *Binary chaos synchronization in elementary cellular automata*, Int. J. Bifurcation Chaos,Volume: 19, Issue: 9, pp. 2871–2884, 2009.
18. R. Dogaru, *Hybrid Cellular Automata as Pseudo-Random Number Generators with Binary Synchronization Property*, in Proceedings of the International Symposium on Signals Circuits and Systems, Iasi Romania, July 2009, pp. 389-392.
19. S. Wolfram, *Random sequence generators*, US patent 4691291A, 1987.
20. D. Das, *A Survey on Cellular Automata and Its Applications*, Global Trends in Computing and Communication Systems, Communications in Computer and Information Science Volume 269, 2012, pp 753–762.
21. R. Dogaru, *Systematic design for emergence in cellular nonlinear networks with applications in natural computing and signal processing*, Springer-Verlag, Berlin Heidelberg, 2008.
22. R. Dogaru, H. Kim, I. Dogaru, *Binary Synchronization in Cellular Automata for Building Compact CDMA Systems*, in Proceedings of the International Sym-posium on Signals Circuits and Systems (ISSCS'09), Iasi Romania, July 2009, pp. 393–396.
23. R. Dogaru, I. Dogaru, H. Kim, *Chaotic Scan: A Low Complexity Video Transmission System for Efficiently Sending Relevant Image Features*, IEEE Trans. on Circuits and Systems for Video Technology, Vol.20, Issue 2, pp. 317–321, 2010.
24. S. Ronjom, M. Abdelraheem and L. E. Danielsen, *TT and ANF Representations of Boolean functions*, in Online Database of Boolean Functions, 2007. Available: http://www.selmer.uib.no/odbf/help/ttanf.pdf
25. I. Dogaru and R. Dogaru, *Algebraic Normal Form for Rapid Prototyping of Elementary Hybrid Cellular Automata in FPGA*, in Proceedings ISEEE 2010 (September 2010, Galati, Romania), pp. 273–276.
26. P. Anghelescu, E. Sofron, S. Ionita, L. Ionescu, *FPGA implementations of cellular automata for Pseudo-random number generation*, International Semiconductor Conference, CAS, 2006, Vol. 2, pp. 371–374.
27. K Cattell and J.Cmuzio. *Synthesis of one-dimensional linear hybrid cellular automata*, IEEE Trans. on Computer-aided design of integrated circuits and systems, 15(3):325–335, 1996.
28. M. S. Turan, *On the nonlinearity properties of maximum-length NFSR feedbacks*, Cryptology ePrint Archive, 2012/112.
29. E. Dubrova, *A Scalable Method for Constructing Galois NLFSRs With Period 2^n-1 Using Cross-Join Pairs*, in IEEE Transactions on Information Theory, (Volume:59, Issue: 1), pp. 703-709, January 2013.
30. Andrei Petrus, (coordinator R. Dogaru), *Random number generator based on cellular automata (FPGA implementation)*, Master Thesis (Romanian language), september 2012, University "Politehnica" of Bucharest.

Received May 20, 2013

THE PUBLISHING HOUSE
OF THE ROMANIAN ACADEMY

PROCEEDINGS OF THE ROMANIAN ACADEMY, Series A,
Volume 14, Special Issue 2013, pp. 373–377

MOBILE SMARTPHONES AS SECURE SIGNATURE-CREATION DEVICES

Adrian FLOAREA[*], Mihai TOGAN[**], Ionut FLOREA[*]

[*] certSIGN, , Bucharest, ROMANIA
[**] Military Technical Academy, Bucharest, ROMANIA
Corresponding author: Adrian FLOAREA, E-mail: adrian.floarea@certsign.ro
Mihai TOGAN, E-mail: mtogan@mta.ro
Ionut FLOREA, E-mail: ionut.florea@certsign.ro

Directive 1999/93/EC of The European Parliament and of the Council on a Community framework for electronic signatures establishes a legal framework for electronic signatures and certain certification-services in order to ensure the proper functioning of the internal market. The legal act defines the specific conditions to be fulfilled by an electronic signature in order to obtain legal recognition. One of the key elements to create electronic signatures with legal value, the advanced electronic signatures, is the usage of secure signature-creation devices. Creation of electronic signatures using a computer is a straightforward process. As mobile devices begin to provide the same services as Personal Computers the creation of electronic signatures is a requirement emerging from the end users. This paper analyses the possibility to create electronic signatures on mobile devices and identifies the challenges: key generation, entropy sources, key management, and protection of the keys on the equipment. It identifies the current possibilities to create advanced electronic signatures on mobile devices and proposes a unified approach to allow the development of a standardized mechanism that turns the smartphones and tablet PCs into secure signature-creation devices.

Key words: electronic signature, key generation, secure element, encryption, smartphones.

1. APPROACH OF SECURITY ISSUES AND SOLUTIONS IN THE MOBILE WORLD

As new technology emerges, new user typologies, mentality shifts and new concerns arise. With the user-centric philosophy it is easy to attract new users and with the always connected paradigm that is even easier to create permanent bonds to pieces of technology, even dependence to the permanent evolution towards complete connectivity.

With the emergence of the new mobile platforms, started with the launch of the first iPhone, the level of information dissemination grew almost exponentially. This means that as we speak, a tremendous, almost unthinkable amount of information travels around us, using various wireless technologies. And much of this information is sensitive at best; part of it is business related info, maybe secret, while another part may be strictly personal.

To secure all this information raises a number of specific problems, related to several aspects:

- The user experience
- The hardware limitations
- The platform limitations
- The difficulty of integrating already developed solutions

By solving the above problems, a robust security solution adapted for mobile platforms will follow some general guidelines:

- Securing the information while in transit
- Securing the information on the device
- Securing the device itself
- Providing the user with tools so that he can tweak and secure his user experience

2. DEVELOPMENT OF SECURITY APPLICATION FOR SMART PHONES

The difference between a PC and a smartphone is that on a smart phone you have to work with much more limited CPU, limited RAM memory and of course, limited SDKs and API.

Developing security application for smart phones requires attention to details as they have a higher risk to be handled by unexpected users if the phone was lost or stolen.

Developing security application on iOS and Android is also a little bit different. Android and iOS are two very different mobile operating systems with very different philosophy. The connection point for compatibility between Android and iOS application can be a portable and well tested crypto library like openSSL [3].

For the implementation of the cryptographic engine on mobile devices there are several options available:

1. Use the Security APIs of each operating system (iOS Security APIs, Android Security APIs, BlackBerry Security APIs, etc.). This approach is recommended to develop applications for a single system and the strongest argument to support it is fast development time.
2. Use the openSSL library. This approach is suitable to develop applications for maximum two mobile operating systems as it is simple to keep compatibility between file format, SMS, emails, etc.
3. Use the openSSL library with a wrapper over openSSL and creating a SDK. This is the best option when developing applications for several mobile operating systems as its provides a unified approach for the software developer. The heaviest part lies upon the developer of the SDK.

To ensure a high degree of cross-platform scalability of a solution, a good common and validated foundation is needed. This kind of foundation is represented by the openSSL libraries, which have many years of development and support on their side, alongside with peer and third party wide support, including applications, infrastructure and backbone. By using openSSL one can be sure that the product of cryptographic operations, by example, obtained on a platform, will be compatible with the solution developed on another openSSL based solution. Also openSSL provides a wide variety of tools, and an entire community that helps to maintain and fix possible bugs and vulnerabilities.

On the other hand, there is no standard build configuration for openSSL to obtain libs especially for iPhone or Android. This means that some effort must be put in adapting the configuration files and the sources so that the libraries can be build, and another effort in testing the results. Also, not all the openSSL modules can be built for mobile platforms, and beside this, between platforms the availability for certain modules differs. As an example, it is more work invested in configuring the build files for iOS, but after that, everything works just fine, as plain C, C++ and objective C can be managed in a unitary way when developing for iPhone.

On Android on the other hand, much more effort must be put in singling out modules that cannot be built for NDK (the Native Development Kit), and then again, additional effort must be put in interfacing the native part of the solution (i.e. the actual engine) by the java side. This implies extended use of JNI (Java Native Interface) as since Android 2.3 native support has been extended to Activity level and solutions based on openSSL can be created and run in an entire native environment.

2.1. Creating electronic signatures on mobile devices

To create electronic signatures on mobile devices it was considered the Option 3 presented above. An SDK, Cryu Base SDK, was developed upon openSSL providing API libraries and developer tools necessary to build, test, and debug security apps for mobile devices. Currently the SDK is available for iOS and Android and the applications developed using the SDK is cross platform interoperable (iOS, Android, PC) as it provides compliance with international standards.

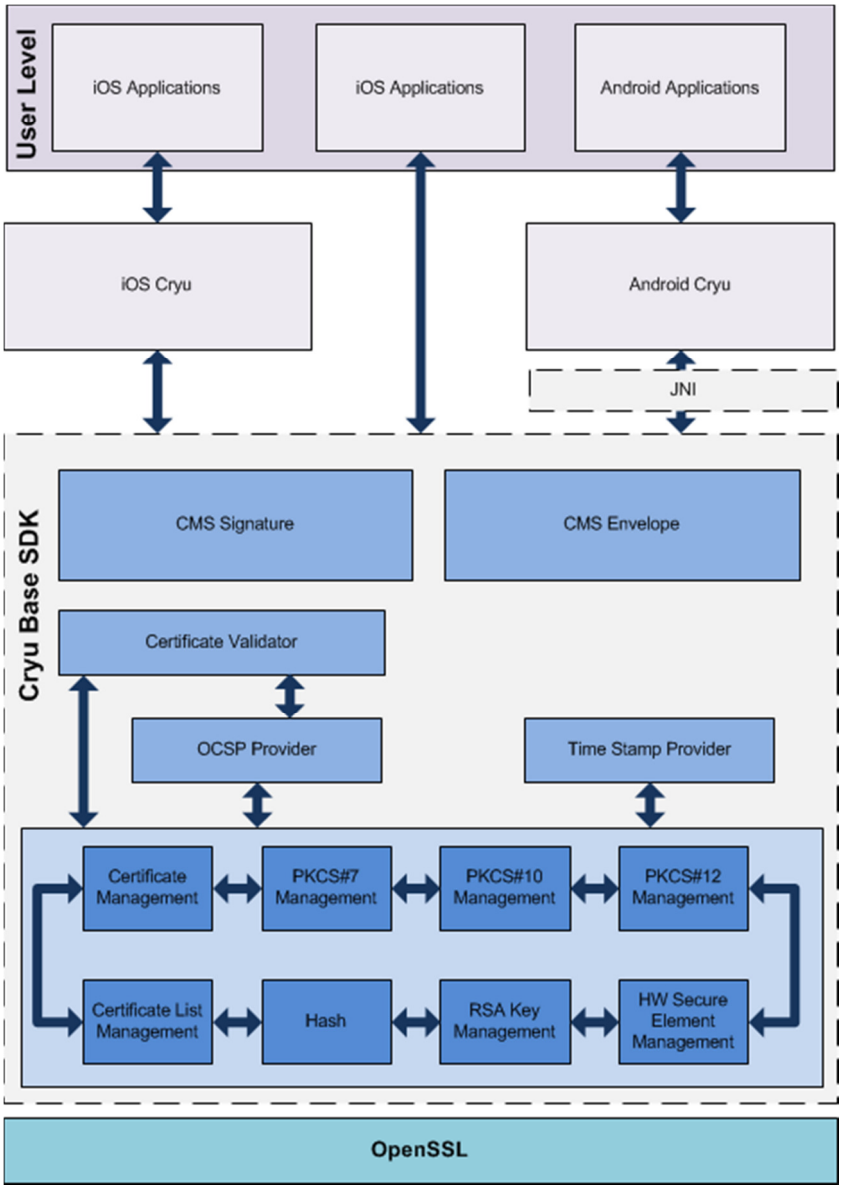


Fig. 1 – SDK Structure.

The structure of the SDK is presented in Fig. 1 and consists of:

- Basic cryptographic functions
  - Key management, hash creation, electronic signature creation, etc.
- Interface with other services: certificate validation, time stamp, etc.
- Interfaces for iOS and Android operating systems

An important element of the SDK is the interface with external secure elements that are generating and storing the cryptographic keys. This allows creation of electronic signature using keys stored on the following secure elements:

- PKCS#11 smart card – Oberthur [5] and Feitian [4]
  - Available only for Android
- Secure SD card with PKCS#11 interface
  - Available only for Android
- SIM card with PKCS#11 interface
  - Available for Android and iOS
- CAC/PIV card – Thurby [6]
  - Available only for iOS

Technically, the electronic signature can be created on mobile devices. Applications that are creating electronic signature were developed using Cryu Base SDK on both Android and iOS. An important constraint is represented by the portability of the secure element, the cross platform interoperability and the ability to be used both on a mobile device and a PC.

From EU Directive 1999/93/EC [1] the security certifications of the secure elements is a key factor for advanced electronic signature creation. An important criteria here is the existence of a security certification for the secure element that is guaranteeing both that the keys were generated correctly, with an acceptable entropy level, that are protected while stored on the device, the access is controlled and the keys cannot be copied. Not all the manufacturers conducted such certifications but, at this moment, it is possible to use several secure elements to protect the keys and create the signature.

Another option investigated was the storage of the keys directly on the mobile device. These offer several entropy sources and built in function to generate random material:

- iOS - `arc4random` function that is also the recommendation from Apple. For a better entropy the following can be added:
  - current system time
  - last values of accelerometer
  - last values of the user touch positions
- Android - the operating system provide access to `/dev/random` and `/dev/urandom`. Due to the fact that mobile operating systems are very rarely shut down, using the /dev/random mechanism is a guarantee that the entropy of our random is a very good one.

The key generation can offer a fair security level but the keys cannot be stored, at this moment, in such way they cannot be copied. This is a major drawback from creating advanced electronic signatures, even if the access to the keys and the security of the device are protected by PIN codes and other configurations that allow even the wipe of the data in the event the mobile device is stolen or an attacker tries a brute force attack over the PIN.

Currently the chip and smart card manufacturers as Intel and Gemalto are working to develop trusted chips to be installed directly on board of the mobile devices. Such chips should allow secure key generation and storage and, in the end, obtain industry certification allowing them to protect keys and create electronic signatures compliant with advanced electronic signature requirements.

3. CONCLUSIONS

The current capabilities of the mobile devices allow users to use them as replacement of the personal Computers and laptops. The demand for electronic signature creation on these devices will emerge in the future and the software manufacturers and hardware producers shall be prepared to answer to this.

Currently there are several technical mechanisms to create electronic signature on mobile devices and some secure elements to store the cryptographic keys are available.

There are differences regarding the interfaces a mobile device provides for connecting a secure element and at this moment a common secure element that can be connected on all equipment using a unique interface does not exist. This is a drawback that reduces portability therefore the keys cannot be used to create electronic signature on different flavors of mobile devices (iOS and Android) as well as on the PC. Several hardware adaptors and connectors are available but they have a limited lifetime as the interfaces are updated frequently.

For the creation of advanced signatures the devices must obtain a security certification, either Common Criteria ELA or FIPS, to guarantee a sound protection of the cryptographic keys.

Currently the manufacturers are conducting research in this area but there is not a strong trend towards migrating electronic signature from the PC to the mobile devices. Cryu Base SDK is an example of cryptographic software development kit that was created and used to develop electronic signature applications on Android and iOS, the mobile platforms with the largest market coverage.

Small market size and the usage of advanced electronic signature only within the boundaries of the European Union is one factor that is doubled also by the adoption of alternative authentication mechanisms, such as OTP. Authentication is widely considered enough secure for the protection of a transaction, even if the integrity of the data is not guaranteed by it.

The adoption of the new EU Regulation [2] on electronic identification and trust services for electronic transactions in the internal market which aims to enhance trust and generate a wider adoption of electronic business can create a larger security market and generate a new momentum for extending the usage of electronic signature. Availability of the mechanisms to create electronic signature using user oriented devices is a key element for the success of this initiative.

REFERENCES

1. Directive 1999/93/EC of the European Parliament and of the Council of 13 December 1999 on a Community framework for electronic signatures: http://europa.eu/legislation_summaries/information_society/other_policies/l24118_en.htm
2. Draft Regulation *on electronic identification and trusted services for electronic transactions in the internal market*: http://ec.europa.eu/information_society/newsroom/cf//itemdetail.cfm?item_id=8544
3. Openssl opensource project, www.openssl.org
4. Feitian Technologies: www.ftsafe.com
5. Oberthur Technologies: www.o**berthur**.com
6. Thursby Software: http://www.thursby.com/

Received July 15, 2013

THE PUBLISHING HOUSE
OF THE ROMANIAN ACADEMY

PROCEEDINGS OF THE ROMANIAN ACADEMY, Series A,
Volume 14, Special Issue 2013, pp. 378–387

ON THE (IN)SECURITY OF GROUP KEY TRANSFER PROTOCOLS BASED ON SECRET SHARING

Ruxandra F. OLIMID

Department of Computer Science, University of Bucharest, Romania
E-mail: ruxandra.olimid@fmi.unibuc.ro

Group Key Transfer (GKT) protocols allow multiple parties to share a common secret key: a trusted Key Generation Center (KGC) selects a uniformly random value that has never been used before and securely distributes it to the legitimate principals. The paper restricts to GKT based on secret sharing; it briefly reviews the security goals and the existing formal security models. We motivate our work by the lack of GKT protocols based on secret sharing that are proved secure in a formal security model. Most of the recent proposals only provide informal and incomplete arguments to claim security, which makes them susceptible to known vulnerabilities. We support our affirmation by exemplifying with three different types of attacks (replay attack, insider attack and known key attack) mounted against protocols published within the last three years. We emphasize that none of these attacks would have been possible against a GKT protocol proved secure in a usual formal security model.

Key words: Group key transfer, Group key establishment, Secret sharing, Formal security model, Replay attack, Insider attack, Known key attack.

1. INTRODUCTION

Group applications have widely spread in the last years. They permit multiple users to benefit of common resources or perform collaborative tasks while they provide differentiate rights or responsibilities within the group. Group applications include text communication, audio, video or web conferences, data sharing or collaborative computing. Let's take the example of a digital conference: multiple users are simultaneously logged in the conference environment. However, they do not all communicate between themselves and should not be aware of the conversations within groups they do not belong to. Usually, the initiator of a conference is a privileged participant who can invite or exclude other parties from the meeting.

Security represents an important aspect for group applications. It is a challenging task to deal with, especially when the group size is large and the members are spread across different (location or networks) areas, with diverse protection mechanisms. In order to obtain the main cryptographic properties as *confidentiality*, *authenticity* and *integrity* it is usually required that the group members previously share a common secret group key. This is achieved as the output of a *group* (or *conference*) *key establishment protocol*.

Group Key Establishment (GKE) protocols divide into [5], [19], [20], [21]: *Group Key Transfer (GKT)* (also called *group key transport* or *group key distribution*) protocols and G*roup Key Agreement (GKA)* (also called *group key exchange*) protocols. In a GKT protocol exists a privileged party named Key Generation Center (KGC) that selects the group key and securely distributes it to the other members. In a GKA protocol each party equally contributes to the key generation, which should not be predetermined by any participant.

This paper focuses on the security of GKT protocols. Next section introduces a short description of GKT and its comparison with GKA, then restricts to GKT protocols based on secret sharing and mentions some of the recent work. Section 3 reviews the informal security goals GKT must achieve and the formal security models designed for GKE (excluding the *contributiveness* goals, which regard only GKA protocols). Section 4 summarises the adversarial and attack types GKT should stand against. Section 5 introduces three examples of attacks on recent introduced GKT protocols. Section 6 concludes.

2. PRELIMINARIES

2.1. Group Key Transfer (GKT)

GKT protocols permit multiple parties to share a common secret group key. The protocol may be executed (concurrently) for multiple times; each execution is called a *session* and it is uniquely identified by a session id. Each session permits a set of *qualified* (also called *authorized* or *legitimate*) participants from the set of registered users to establish the common key, which should remain hidden for any other party. A *registered* user is a party who has previously registered to the KGC and with whom shares a long-lived key (usually a public-secret key pair the user uses for signing).

The session key is chosen by the KGC, which must be trusted by all participants as honest in the sense that it selects a *fresh* key (a uniformly random chosen value that has never been used before). This trust assumption is not required for GKA, which do not demand the existence of a privileged party to select the key, but compute it by equal contribution of all principals. However, regardless of the GKE type, a trust relation is mandatory: the qualified participants to a session trust each other that none of them discloses the shared key. Otherwise, the confidentiality of the protocol is violated by default.

Due to the fact that parties do not necessary have to communicate between themselves (but only with the KGC), the computational and transmission costs of GKT protocols are usually lower that those of GKA protocols. Also, the design of GKT is in general less challenging.

Depending on the application needs or constraints (security requirements, computational resources and transmission costs), a GKT or a GKA protocol may suit best.

A general mechanism for defining GKT protocols is immediate [7]: KGC generates a fresh group key and sends its encrypted value under the appropriate key to each legitimate participant. Hence, any authorized user decrypts and finds the key, while it remains secure against unauthorized parties. We have assumed that an authentication mechanism exists, such that the KGC or the users cannot be impersonated and the message cannot be modified during transmission. This trivial solution becomes inefficient for large groups: KGC must perform m encryptions and send m messages, where m is the number of qualified participants. In case a symmetric encryption scheme is used to decrease the computational costs (rather than an asymmetric encryption scheme), a supplementary assumption appears: each registered group member must previously share a secret with the KGC.

Secret sharing is used in GKE protocols to avoid such disadvantages; in addition, they introduce several benefits: a convenient way to differentiate between principals power within the group, delegation of duties by passing shares to other participants, group authentication instead of entity authentication, cheating detection and simple management of group sizing using the accepted threshold [26]. Next section briefly reminds secret sharing.

2.2. Secret Sharing

Blackley [4] and Shamir [27] independently introduced secret sharing as a solution to the key management problem. It represents a mechanism that splits a secret into multiple shares such that the secret may be recovered only from authorized sets of shares. In general, a secret sharing scheme consists of three phases: *sharing* (a dealer splits the secret into multiple parts, called *shares*), *distribution* (the dealer securely sends one or more shares to each party) and *reconstruction* (a qualified set of parties combine their shares to recover the secret).

The set of all authorized sets of shares in called the *access structure*. A secret sharing scheme whose access structure contains all sets with at least t out of m shares is called (t,m) *threshold secret sharing scheme*; a secret sharing scheme whose access structure contains only the set of all shares is called *all-or-nothing* (or *unanimous*) secret sharing scheme.

Various GKE protocols based on secret sharing exist in the literature. We only remind here some of the recent work in the field of GKT protocols: Harn and Lin's protocol based on Shamir's scheme (2010) [15], Hsu *et al.*'s protocol based on linear secret sharing (2012) [16], Sun *et al.*'s protocol based on derivative secret sharing (2012) [28] and Yuan *et al.*'s protocol based on Shamir's scheme (2013) [30]. We highlight

that even though all mentioned protocols were published in the last three years, none of them gives a security proof in a formal model, but rather incomplete and informal arguments of security. This leads to a high opportunity to reveal vulnerabilities. We emphasize that the existing work in the field of GKE protocol based on secret sharing scheme with formal security proofs is very limited.

3. SECURITY DEFINITIONS AND MODELS

3.1. Informal Security Definitions

A GKT protocol can be considered secure if it achieves the properties that we informally recall next [19], [20]. We restrict to security goals that apply to GKT and do not remind specific properties of GKA, which are beyond the scope of this paper (such as *key contributiveness* or *key control*).

Key confidentiality (also called *key privacy*, *key secrecy* or *non-disclosure*) [7], [12], [13], [17] guarantees that it is (computationally) infeasible for an adversary to compute the group key. The stronger notion of *known key security* [6], [29] assures that key confidentiality is maintained even if the attacker somehow manages to obtain group keys of previous sessions. *Backward secrecy* [11], [20] generalizes this concept and conserves the privacy of future keys regardless the adversary's actions in the past sessions. Correspondingly, *forward secrecy* [11], [13] imposes that the adversary actions in future runs of the protocol do not compromise the privacy of previous session keys (*i.e.* a key remains secure in the future).

Key selection must satisfy specific properties. *Key freshness* [21] requires that the group key is new (*i.e.* it has never been used before). The related concept of *key independence* [17], [23] imposes that no correlation exists between keys from different sessions; this means that (cooperation between) authorized participants to distinct sessions of the protocol cannot disclose session keys they are unauthorized for. In addition, *key randomness* warrants *key indistinguishability* from a random number and hence *key unpredictability*. Two other important security requirements regarding the key value exists: *key integrity* [17], which attests that no adversary can modify the group key and *key consistency,* which prevents different players to accept different keys.

Group member authentication represents a mandatory condition for group cryptographic protocols. *Entity authentication* [1] confirms the identity of a participant to the others. Similarly, *unknown key share resilience* [13] restricts a user to believe that the key is shared with one party when in fact it is shared with another. *Key compromise impersonation resilience* [3], [14] prevents an attacker who owns the long-lived key of a participant to impersonate other parties to him. The stronger property named *ephemeral key leakage resilience* [31] avoids an adversary to recover the group key even if he discloses the long-lived keys and ephemeral keys of parties involved except both these values for the participants in the test session.

(Implicit) Key authentication [21] limits the possible owners of the group key to the legitimate participants; this means that no other party except the qualified users is capable to compute the key, but it does not necessary mean that all legitimate principals actually own it. Another property, called *key confirmation* [5], [21] certifies that all authorized members actually have the key; however, it does not claim that no other party own the same key. *Explicit key authentication (or Mutual Authentication)* [2], [8], [11], [21] combines these notions and ensures that all qualified participants to the protocol have actually computed the group key and no one else except them has.

For more information on informal security requirements, we invite the reader to refer to [19] and [20].

3.2. Formal Security Models

Security models formalize the properties described in the previous subsection within a precise environment, specifying the trust assumptions, the relations between participants, the adversarial power, the communication medium and others relevant aspects. Similar to the informal definitions mentioned before, GKE security models were developed as a generalization of the existing two or three party security models.

Bresson *et al.* introduced the first security model for GKE protocols in 2001 [10]. Their model was rapidly extended to allow dynamic groups, meaning that the group membership may change during the protocol execution [8]. One year later, the same authors improved their model to stand against strong

corruptions, which permit an attacker to reveal the ephemeral internal state information of the user instances [9]. In 2005, Katz and Shin considered the existence of malicious users, in the sense that even if a registered party always knows the key of sessions he is qualified for, he must be restricted to perform malicious actions [18]; for example, he should not disclose session keys he is unauthorized for, modify the value of the common key as he desires or make honest users compute different keys. Recently, stronger security notions were considered: Gorantla, Boyd and González Nieto introduced in 2009 a model that deals with key compromise impersonation [14], while Zhao *et al.* enhanced it in 2011 to achieve ephemeral key leakage resilience [31]. For an extensively survey on group key security models, the reader may address to [19] and [20].

We skip the formal definitions of the security models, as they do not represent the goal of this paper. However, we briefly remind that they rely on two main security requirements known as *AKE-Security* (*Authenticated Key Exchange Security*) and *MA-Security* (*Mutual Authenticated Security*). AKE-Security's main objective is to prevent unqualified members to reveal the common group key; it aims informal security notions such as key confidentiality, forward and backward secrecy, known key security, key compromise impersonation or ephemeral key leakage resilience. MA-Security's primary goal is to make users aware of the correct identity of the other parties and compute identical keys at the end of the protocol; it unifies informal concepts such as (implicit) key authentication, key confirmation or unknown key share resilience. We bypass the notion of *key contributiveness*, which only applies to GKA protocols.

Although much research has been done in the last years in the field, the security models still have some limitations concerning the adversarial capabilities. For example, the existing models do not deal with DoS (Denial of Service) or fault tolerance [11]. Protocols remain vulnerable to this kind of attacks, even if they are proved to be secure in formal security models.

4. ADVERSARIES AND ATTACKS CLASSIFICATION

A secure GKT protocol must stand against *passive* and *active* adversaries. A passive adversary can only eavesdrop on the communication channel, while an active adversary has full control over the network (he can drop, modify or insert messages). It is immediate that an active attack is more powerful than a passive attack and therefore active attacks should be considered when formally analysing the security of a group key protocol.

Regarding the appartenance to the group, attackers split into: *outsiders* and *insiders*. An outsider is a party that has not registered as a group member (and hence does not posses a valid long-lived key within the group) and never takes part to the key establishment as a legitimate participant. An outsider attack aims to reveal the established group key and therefore to break the AKE-Security of the protocol, usually by impersonating authorized users. An insider is a valid group member, who has registered within the group at a given moment and therefore has the advantage to posses a long-lived key. He is qualified to compute session keys he is authorized for, but this should provide him no advantage in revealing other keys (of sessions he is unqualified for) or damage the protocol in any other way: find the long-term keys of other users, ruin key consistency or get control over the key value. Of course, insiders are more powerful than outsiders, because they have access to additional information. Within the definition of formal security models, an adversary is not considered to be a malicious user, but an external attacker who has access to the long-lived keys and/or ephemeral values used during the run of the protocol by making queries [11]. We illustrate an insider attack [25] on Yuan *et al.*'s protocol [30] later in this paper.

Impersonation attacks try to make messages originating from the adversary indistinguishable from messages originating from legitimate users (the adversary pretends to be a qualified group member). This may result in computing a different key than the genuine one or in establishing a common key with an attacker instead of an authorized user. A successful impersonation attack can for example break entity authentication, unknown key share resilience or key compromise impersonation resilience. A special kind of attack is the *replay attack* that consists in injecting messages from previous executions of the protocol. This can turn into a *key replication attack*, where the same (corrupted) key is derived for multiple runs of the protocol. It is immediate that a key replication attack violates key freshness. We review a replay attack [22] on Harn and Lin's construction [15] in Section 5.

Known key attacks aim to disclose a session key when the adversary knows at least one key from a previous run of the protocol. All insiders satisfy the assumption by default, as they may legitimate take part to protocol executions. We exemplify a known key attack [24] on Sun *et al.*'s proposal [28] in the next section.

Attacks can be classified based on the information the adversary has access to: the long-lived key of the registered users or the ephemeral secrets used during protocol execution. *Opening attacks* allow the attacker to learn the ephemeral secrets without revealing the long-lived secrets, while *weak corruption attacks* allow the attacker to learn the long-lived secrets without revealing ephemeral secrets. *Strong corruption attacks* combine these two attacks and give the adversary tremendous power: he can corrupt a user and obtain both his long-lived secret and his ephemeral secret values [11].

DoS (Denial of Service) attacks lead to the futility of GKT protocols: they prevent legitimate users to establish common secret keys that they would have later use for application purposes. A DoS attack may inhibit users to compute any key at all or may force users to end up with different keys. Although the attack is discovered at the latest during the execution of the application (the group members realise that they cannot properly communicate between themselves) it represents an important aspect of network security. In contrast to the previously mentioned attacks, we highlight that the DoS attacks are not considered within the existing formal security models for GKE protocols [11].

In formal security models, the adversary is modelled as a PPT (Probabilistic Polynomial Time) algorithm with full control over the communication channel (hence active in the sense that it can inject, delete or modify exchanged messages). He interacts with the legitimate group members by asking queries with the scope to reveal information that he may use in breaking the AKE-Security or MA-Security. For example, he makes Corrupt queries to obtain the long-lived key of registered participants, RevealState queries to retrieve ephemeral secret used by parties during the protocol execution or RevealKey queries to retrieve the common shared key of particular sessions. Each security model defines the queries an attacker is allowed to make (which model the power of the attacker) as well as the security game he plays against the AKE-Security or MA-Security of the protocol (which he breaks if he wins the game with non-negligible probability). We will skip the formal definitions of particular formal models, but strongly invite the reader to address some of the recent original papers [11], [14], [31] or surveys [19], [20].

5. ATTACKS ON RECENT GKT PROTOCOLS

The current section describes three types of attacks against recent GKT protocols: a replay attack on Harn and Lin's protocol [15] introduced by Nam *et al.* [22], an insider attack on Yuan *et al.*'s proposal [30] defined by Olimid [25] and a known key attack on Sun *et al.*'s construction [28] revealed by Olimid [24]. We emphasize that all mentioned protocols lack a formal security proof and hence the vulnerabilities arise natural.

For the rest of the paper, we will use the following notations: m the number of possible users, $\{U_1,...,U_t\}, t \le m$ the set of participants to a given session (after reordering) with U_1 as initiator, h, h_1 and h_2 collision-resistant hash functions, $\leftarrow^R$ a random choice from a specified set of values, $||$ string concatenation, $(s_j), j = 1..4$ specific protocol sessions.

Let U_a be the attacker. His main goal is to break the AKE-Security or MA-Security of the protocol. U_a may be an insider ($U_a \in \{U_1,...,U_m\}$) or an outsider ($U_a \notin \{U_1,...,U_m\}$), depending the adversarial scenario.

We proceed with the description of the protocols and the attacks. For more details, we invite the reader to address the original papers.

Protocol 1. Harn and Lin [15]

Initialization. KGC selects two large safe primes p and q and computes $n = pq$.

Users Registration. Each user $U_i, i = 1..m$ shares a long-lived secret $(x_i, y_i) \in Z_n^* \times Z_n^*$ with the KGC.

Round 1. User U_1 sends a key generation request:

$$U_1 \to KGC : (U_1,...,U_t).$$

Round 2. KGC broadcasts the list of participants as a response:

$$KGC \to^* : (U_1,...,U_t).$$

Round 3. Each user $U_i, i = 1..t$ chooses $r_i \leftarrow^R Z_n^*$, computes $Auth_i = h(x_i, y_i, r_i)$ and sends:

$$U_i \to KGC : (r_i, Auth_i).$$

Round 4. KGC checks if $Auth_i = h(x_i, y_i, r_i), i = 1..t$ (otherwise he aborts), selects a group key $k \leftarrow^R Z_n^*$, generates the polynomial $f(x)$ of degree t that passes through $(0,k)$ and $(x_i, y_i \oplus r_i), i = 1..t$, computes t additional points $P_1,...,P_t$ on $f(x)$ and the authentication message $Auth = h(k, U_1,...,U_t, r_1,...,r_t, P_1,...,P_t)$, then broadcasts:

$$KGC \to^* : (P_1,...,P_t, Auth).$$

Key Computation. Each user $U_i, i = 1..t$ computes the group key $f(0)$ by interpolating the points $P_1,..,P_t$ and $(x_i, y_i \oplus r_i)$ and checks if $Auth = h(k, U_1,...,U_t, r_1,...,r_t, P_1,...,P_t)$ (otherwise he aborts).

Attack 1. Replay Attack [22]

Attack scenario. U_a is an insider whose goal is to break the AKE-Security of Protocol 1: he obtains the key of any session a user $U_i \in \{U_1,..,U_{a-1},U_{a+1},..,U_m\}$ is qualified for (even if U_a is unqualified for) by disclosing the long-lived secret of U_i.

Step 1. U_a eavesdrops $(r_i, Auth_i)$ from a session U_i is qualified for.

Step 2. U_a initiates two legitimate sessions of the protocol with U_i, denoted by $(s_j), j = 1,2$.

Step 3. In both sessions, U_a impersonate the legitimate user U_i by sending the eavesdropped message $(r_i, Auth_i)$ and behaves honestly in sending his own message $(r_a, Auth_a)$.

Step 4. U_a recovers the coefficients of the polynomials $f(x)_{(s_j)} = a_{(s_j)}x^2 + b_{(s_j)}x + c_{(s_j)}, j = 1,2$, as being a qualified participant for sessions $(s_j), j = 1,2$.

Step 5. As $(x_i, y_i \oplus r_i)$ and $(x_a, y_a \oplus r_a)$ are valid points on $f(x)_{(s_j)}, j = 1,2$, x_i and x_a are two roots of the quadratic equation:

$$(a_{(s_1)} - a_{(s_2)})x^2 + (b_{(s_1)} - b_{(s_2)})x + c_{(s_1)} - c_{(s_2)} = 0. \quad (1)$$

Step 6. U_a computes the secret key of U_i as:

$$\begin{aligned} x_i &= x_a^{-1}(a_{(s_1)} - a_{(s_2)})^{-1}(c_{(s_1)} - c_{(s_2)}) \\ y_i &= f(x_i)_{(s_1)} \oplus r_1 = f(x_i)_{(s_2)} \oplus r_2. \end{aligned} \quad (2)$$

Step 7. U_a discloses all keys of the sessions U_i is qualified for by using the long-lived secret (x_i, y_i).

Protocol 2. Yuan *et al.* [30]

Initialization. KGC selects two large primes p and q and computes $n = pq$.

Users Registration. Each user $U_i, i = 1..m$ shares a long-lived secret password $pw_i = pw_{ix} \| pw_{iy}$ with the KGC.

Round 1. User U_1 chooses $k_1 \leftarrow^R Z_n$, computes $K_1 = pw_{1x} + k_1$ and $M_1 = h_1(U_1,...,U_t, k_1)$, then sends a key generation request:

$$U_1 \to KGC : (U_1, \{U_1,...,U_t\}, K_1, M_1).$$

Round 2. KGC computes $k_1 = K_1 - pw_{1x}$, checks if $M_1 = h_1(U_1,..,U_t,k_1)$ (otherwise he aborts) and broadcasts the list of participants as a response:

$$KGC \to^* : (U_1,...,U_t).$$

Round 3. Each user $U_i, i = 2..t$ chooses $k_i \leftarrow^R Z_n$, computes $K_i = pw_{ix} + k_i$ and $M_i = h_i(U_1,..,U_t,k_i)$, then sends:

$$U_1 \to KGC : (U_i, \{U_1,...,U_t\}, K_i, M_i).$$

Round 4. KGC computes $k_i = K_i - pw_{ix}$, checks if $M_i = h_i(U_1,..,U_t,k_i), i = 2,..,t$ (otherwise he aborts), selects 2 random numbers x_{ta} and y_{ta} of lengths equal to pw_{ix} and pw_{iy}, generates the polynomial $f(x)$ of degree t that passes through (x_{ta}, y_{ta}) and $(pw_{ix}, pw_{iy} + k_i), i = 1..t$, computes t additional points $P_1,...,P_t$ on $f(x)$ and the verification messages $V_i = h_2(U_1,...,U_t,P_1,..,P_t,k_i), i = 1..t$, then sends:

$$KGC \to^* : (P_1,...,P_t,V_i).$$

Key Computation. Each user $U_i, i = 1..t$ checks if $V_i = h_2(U_1,...,U_t,P_1,..,P_t,k_i)$ (otherwise he aborts) and computes the group key $k = f(0)$ by interpolating the points $P_1,...,P_t$ and $(pw_{ix}, pw_{iy} + k_i)$.

Attack 2. Insider Attack [25]

Attack scenario. U_a is an insider whose goal is to break the AKE-Security of Protocol 2: he obtains the key of any session a user $U_i \in \{U_1,..,U_{a-1},U_{a+1},..,U_m\}$ is qualified for (even if U_a is unqualified for) by disclosing the long-lived secret password of U_i.

Step 1. U_a initiates four legitimate sessions of the protocol with U_i, denoted by $(s_j), j = 1..4$.

Step 2. U_a recovers the coefficients of the polynomials $f(x)_{(s_j)} = a_{(s_j)}x^2 + b_{(s_j)}x + c_{(s_j)}, j = 1..4$, as being a qualified participant for sessions $(s_j), j = 1..4$.

Step 3. U_a eavesdrops the values $K_{i(s_j)}, j = 1..4$; as $(pw_{ix}, pw_{iy} + k_{i(s_j)})$ are valid points on $f(x)_{(s_j)}, j = 1..4$ and $K_{i(s_j)} = pw_{ix} + k_{i(s_j)}$, U_a obtains:

$$pw_{iy} = a_{(s_j)} pw_{ix}^{\ 2} + (b_{(s_j)} + 1)\, pw_{ix} + c_{(s_j)} - K_{i(s_j)}, j = 1..4. \tag{3}$$

Step 4. U_a eliminates pw_{iy} from the first two equations ($j = 1,2$), respectively from the last two equations ($j = 3,4$) in (3) and obtains:

$$\begin{cases} A_{(s_1,s_2)} pw_{ix}^{\ 2} + B_{(s_1,s_2)}\, pw_{ix} + C_{(s_1,s_2)} = 0 \\ A_{(s_3,s_4)} pw_{ix}^{\ 2} + B_{(s_3,s_4)}\, pw_{ix} + C_{(s_3,s_4)} = 0, \end{cases} \tag{4}$$

where:

$$\begin{aligned} A_{(s_1,s_2)} &= a_{(s_1)} - a_{(s_2)} & A_{(s_3,s_4)} &= a_{(s_3)} - a_{(s_4)} \\ B_{(s_1,s_2)} &= b_{(s_1)} - b_{(s_2)} \quad ; & B_{(s_3,s_4)} &= b_{(s_3)} - b_{(s_4)} \\ C_{(s_1,s_2)} &= c_{(s_1)} - c_{(s_2)} - (K_{i(s_1)} - K_{i(s_2)}) & C_{(s_3,s_4)} &= c_{(s_3)} - c_{(s_4)} - (K_{i(s_3)} - K_{i(s_4)}) \end{aligned} \tag{5}$$

Step 4. U_a computes the secret key of U_i as:

$$pw_{ix} = (A_{(s_1,s_2)}C_{(s_3,s_4)} - A_{(s_3,s_4)}C_{(s_1,s_2)})(A_{(s_3,s_4)}B_{(s_1,s_2)} - A_{(s_1,s_2)}B_{(s_3,s_4)})^{-1}$$
$$pw_{iy} = f(pw_{ix})_{(s_j)} + pw_{ix} - K_{i(s_j)}, j = 1..4. \quad (6)$$

Step 5. U_a discloses all group keys of the sessions the user U_i is qualified for by using the long-lived password $pw_i = pw_{ix} \| pw_{iy}$.

Protocol 3. Sun *et al.* [28]

Initialization. KGC selects a multiplicative cyclic group G of prime order p with g as generator.

Users Registration. Each user $U_i, i = 1..m$ shares a long-lived secret $s_i' \in G$ with the KGC.

Round 1. User U_1 sends a key generation request:

$$U_1 \to KGC : (U_1,...,U_t).$$

Round 2. KGC broadcasts the list of participants as a response:

$$KGC \to^* : (U_1,...,U_t).$$

Round 3. Each user $U_i, i = 1..t$ chooses $r_i \leftarrow^R Z_p^*$ and sends it to the KGC:

$$U_i \to^* r_i.$$

Round 4. KGC selects a value $S \leftarrow^R G$, invokes the derivative secret sharing scheme to split S into 2 parts t times such that $S = s_i + s_i'$ (in G), $i = 1,..,t$, computes the session group key as $k = g^S$ (in G), t messages $M_i = (g^{s_i+r_i}, U_i, h(U_i, g^{s_i+r_i}, s_i', r_i)), i = 1..t$ and $Auth = h(k, g^{s_1+r_1},..., g^{s_t+r_t}, U_1,...,U_t, r_1,..., r_t)$, then broadcasts:

$$KGC \to^* : (M_1,..., M_t, Auth).$$

Key Computation. Each user $U_i, i = 1..t$ checks if $h(U_i, g^{s_i+r_i}, s_i', r_i)$ equals the corresponding value in M_i (otherwise he aborts), computes $k = g^{s_i'} g^{s_i+r_i} (g^{r_i})^{-1}$, checks if $Auth = h(k, g^{s_1+r_1},..., g^{s_t+r_t}, U_1,...,U_t, r_1,..., r_t)$ (otherwise he aborts), accepts k as the group key and sends:

$$U_i \to KGC : h_i = h(s_i', k, U_1,...,U_t, r_1,..., r_t).$$

Key Confirmation. KGC checks if $h_i = h(s_i', k, U_1,...,U_t, r_1,..., r_t)$ using his knowledge on s_i' and k, certifying that all users posses the same key.

Attack 3. Known Key Attack [24]

Attack scenario. U_a is an adversary who knows the key $k_{(s_1)}$ of a session (s_1) and his goal is to break the AKE-Security of Protocol 3: he obtains the key $k_{(s_2)}$ of another session (s_2) (he is unauthorized for) that has at least one common qualified participant U_i with (s_1).

Step 1. U_a eavesdrops $r_{i_{(s_j)}}$ and $g^{s_{i(s_j)}+r_{i(s_j)}}, j = 1,2$ and computes:

$$g^{s_{i(s_j)}} = g^{s_{i(s_j)}+r_{i(s_j)}} (g^{r_{i(s_j)}})^{-1}, j = 1,2. \quad (7)$$

Step 2. U_a knows the session key $k_{(s_1)}$ and uses the previously computed value $g^{s_{i(s_1)}}$ to obtain:

$$g^{s_i'} = k_{(s_1)} (g^{s_{i(s_1)}})^{-1} \quad (8)$$

Step 3. U_a recovers the session key $k_{(s_2)}$ from (7) and (8) as:

$$k_{(s_2)} = g^{s_{i(s2)}} g^{s_i} \tag{9}$$

6. CONCLUSIONS

The paper considers GKT protocols based on secret sharing schemes. It provides a brief survey on the informal security notions that GKT protocols should satisfy and reviews the adversarial models and types of attacks that GKT protocols should stand against. Security models formalize such requirements within a specific environment. We highlight the necessity of electing a protocol secure within a suitable model, considering the best trade-off between efficiency and the required security level for each application.

Unfortunately, current work on GKT protocols based on secret sharing neglect this aspect. Recent published protocols ignore formal security and only rely on incomplete and informal arguments. This makes them susceptible to known vulnerabilities that would have been excluded by a proper proof under a convenient security model. In order to support our claim, we recall three different types of attacks against GKT protocols using secret sharing that were published within the last three years: a replay attack, an insider attack and a known key attack. We mention that all these attacks are considered, and therefore impossible to succeed, in all usual security models.

Although much work has been done in the field of formal security lately, we must admit some limitations. The GKE formal security models do not deal with all known vulnerabilities. For example, they do not stand against DoS attacks. Therefore, even if a GKT protocol is proved secure in a strong security model, it can become useless if participants are restricted to compute the session key. This can be easily achieved by blocking the messages to arrive to the qualified participants: since a user misses mandatory information, he can no longer recover the session key and therefore halts. We admit that DoS attacks are probably impossible to handle only by cryptographic techniques and therefore require proper protection mechanisms.

To conclude, we highlight the problems a protocol without a formal security proof may raise and emphasize the deficiency of provable secure GKT protocols based on secret sharing in the literature, which we consider a direction for further research.

ACKNOWLEDGEMENTS

This paper is supported by the Sectorial Operational Program Human Resources Development (SOP HRD), financed from the European Social Fund and by the Romanian Government under the contract number SOP HDR/107/1.5/S/82514.

REFERENCES

1. M. Bellare, P. Rogaway, *Entity Authentication and Key Distribution,* In Advances in Cryptology (CRYPTO'93), pp. 232–249, 1993.
2. M. Bellare, P. Rogaway, *Random Oracles are Practical: A Paradigm for Designing Efficient Protocols*, In Proceedings of the 1st ACM Conference on Computer and Communications Security (CCS'93), pp. 62–73, 1993.
3. S. Blake-Wilson, D. Johnson, A. Menezes, *Key Agreement Protocols and Their Security Analysis*, In Proceedings of the 6th IMA International Conference on Cryptography and Coding, pp. 30–45, 1997.
4. G. Blakley, *Safeguarding Cryptographic Keys,* In Proceedings of the AFIPS'79, pp. 313–317, 1979.
5. C. Boyd, *On Key Agreement and Conference Key Agreement*, In Information Security and Privacy: Australasian Conference, pp. 294–302, 1997.
6. M. Burmester, *On the Risk of Opening Distributed Keys*, In Advances in Cryptology (CRYPTO'94), pp.308–317, 1994.
7. M. Burmester, Y. Desmedt, *A Secure and Efficient Conference Key Distribution System,* In Proceedings of EUROCRYPT'94, pp. 275–286, 1994.
8. E. Bresson, O. Chevassut, D. Pointcheval, *Provably Authenticated Group Diffie-Hellman Key Exchange - The Dynamic* Case, In Proceedings of ASIACRYPT'01, pp. 290–309, 2001.
9. E. Bresson, O. Chevassut, D. Pointcheval, *Dynamic Group Diffie-Hellman Key Exchange under Standard Assumptions*, In Proceedings of EUROCRYPT'02, pp. 321–336, 2002.

10. E. Bresson, O. Chevassut, D. Pointcheval, J.J. Quisquater, *Provably Authenticated Group Diffie-Hellman Key Exchange*, In Proceedings of the 8th ACM Conference on Computer and Communications Security (CCS'01), pp. 255–264, 2001.
11. E. Bresson, M. Manulis, *Securing Group Key Exchange against Strong Corruptions*, In Proceedings of ASIACSS'08, pp. 249–260, 2008.
12. W. Diffie, M.E. Hellman, *New Directions in Cryptography*, IEEE Transactions on Information Theory, IT-**22**(6), pp. 644–654, 1976.
13. W. Diffie, P.C. van Oorschot, M.J. Wiener, *Authentication and Authenticated Key Exchanges*, Designs, Codes and Cryptography, **2**(2), pp. 107–125, 1992.
14. M.C. Gorantla, C. Boyd, J.M. González Nieto, *Modeling Key Compromise Impersonation Attacks on Group Key Exchange Protocols,* In Proceedings of Public Key Cryptography (PKC'09), pp. 105–123, 2009.
15. L. Harn, C. Lin, *Authenticated Group Key Transfer Protocol based on Secret Sharing*, IEEE Trans. Comput. **59**(6), pp. 842–846, 2010.
16. C. Hsu, B. Zeng, Q. Cheng, G. Cui, *A Novel Group Key Transfer Protocol*, Cryptology ePrint Archive, Report 2012/043, 2012.
17. P. JANSON, G. TSUDIK, *Secure and Minimal Protocols for Authenticated Key Distribution*, Computer Communications, **18**(9), pp. 645–653, 1993.
18. J. Katz, J.S. Shin, *Modeling Insider Attacks on Group Key-Exchange Protocols*, In Proceedings of the 12th ACM Conference on Computer and Communications Security (CCS'05), pp. 180–189, 2005.
19. M. Manulis, *Survey on Security Requirements and Models for Group Key Exchange*, Technical Report 2006/02, 2006.
20. M. Manulis, *Provably Secure Group Key Exchange*, European University Press, 2007.
21. A. Menezes, P. van Oorschot, S. Vanstone, *Handbook of Applied Cryptography*, CRC Press, 1996.
22. J. Nam, M. Kim, J. Paik, W. Jeon, B. Lee, D. Won, *Cryptanalysis of a Group Key Transfer Protocol based on Secret Sharing*, In Proceedings of the 3rd International Conference on Future Generation Information Technology, pp. 309–315, 2011.
23. M. Steiner, G. Tsudik, M. Waidner, *CLIQUES: A New Approach to Group Key Agreement,* In Proceedings of the 18th International Conference on Distributed Computing Systems (ICDCS'98), pp. 380–387, 1998.
24. R.F. Olimid, *On the Security of an Authenticated Group Key Transfer Protocol Based on Secret Sharing*, In Proceedings of ICT-EurAsia 2013 (AsiaARES 2013), pp. 399–408, 2013.
25. R.F. Olimid, *Cryptanalysis of a Password-based Group Key Exchange Protocol Using Secret Sharing*, Appl. Math. Inf. Sci.**7**(4), pp. 1585–1590, 2013.
26. J. Pieprzyk, C.H. Li, *Multiparty Key Agreement Protocols*, In IEEE Proceedings - Computers and Digital Techniques, pp. 229–236, 2000.
27. A. Shamir, *How to Share a Secret*, Commun. ACM **22**(11), pp. 612–613, 1979.
28. Y. Sun, Q. Wen, H. Sun, W. Li, Z. Jin, H. Zhang, *An Authenticated Group Key Transfer Protocol based on Secret Sharing,* Procedia Engineering **29**(0), pp. 403–408, 2012.
29. Y. Yacobi, Z. Shmuely, *On Key Distribution Systems*, In Advances in Cryptology (CRYPTO'89), pp. 344–355, 1990.
30. W. Yuan, L. Hu, H. Li, J. Chu, *An Efficient Password-based Group Key Exchange Protocol Using Secret Sharing*, Appl. Math. Inf. Sci. **7**(1), pp. 145–150, 2013.
31. J. Zhao, D. Gu, M.C. Gorantla, *Stronger Security Model of Group Key Agreement*, In Proceedings of the 6th ACM Symposium on Information, Computer and Communications Security (ASIACCS'11), pp. 435–440, 2011.

Received June 1, 2013