Best Effort and Practice Activation Codes

Gerhard de Koning Gans[1] and Eric R. Verheul[1,2]

[1] Institute for Computing and Information Sciences
Radboud University Nijmegen
P.O. Box 9010, 6500 GL Nijmegen, The Netherlands
{gkoningg,eric.verheul}@cs.ru.nl
[2] PricewaterhouseCoopers Advisory
P.O. Box 22735, 1100 DE Amsterdam, The Netherlands
eric.verheul@nl.pwc.com

Abstract. Activation Codes are used in many different digital services and known by many different names including voucher, e-coupon and discount code. In this paper we focus on a specific class of ACs that are short, human-readable, fixed-length and represent value. Even though this class of codes is extensively used there are no general guidelines for the design of Activation Code schemes. We discuss different methods that are used in practice and propose BEPAC, a new Activation Code scheme that provides both authenticity and confidentiality. The small message space of activation codes introduces some problems that are illustrated by an adaptive chosen-plaintext attack (CPA-2) on a general 3-round Feistel network of size 2^{2n}. This attack recovers the complete permutation from at most 2^{n+2} plaintext-ciphertext pairs. For this reason, BEPAC is designed in such a way that authenticity and confidentiality are independent properties, i.e. loss of confidentiality does not imply loss of authenticity.

Keywords: activation code, e-coupon, voucher, Feistel network, small domain encryption, financial cryptography.

1   Introduction

This paper introduces Activation Codes (ACs) as a generic term for codes that are used in many different digital services. They are known by many different names including voucher, e-coupon and discount code. The common properties of these codes are that they need to be short, human-readable, have a fixed length and can be traded for economic benefit. There are schemes [4,5,8] that include all kinds of property information in the code itself or include digital signatures [14,12]. This makes the codes unsuitable for manual entry and thus for printing on products, labels or receipts. The focus of this paper is on ACs that can be printed and manually entered such as the AC that is printed on a receipt in Figure 1. In this case the customer can enter the AC 'TY5FJAHB' on a website to receive some product. We propose a scheme called BEPAC to generate

and verify this class of ACs. BEPAC is an acronym for Best Effort and Practice Activation Codes. Here '*best practice*' covers the use of a keyed hash function to satisfy authenticity and '*best effort*' covers the use of a Feistel network to satisfy confidentiality.

Security plays an important role in the design of an AC system because of the economic value it represents. A system breach could result in big financial losses. Nevertheless, to the best of our knowledge, there are no guidelines on the design of secure AC systems that consider the previously mentioned properties. Despite the lack of general guidelines for good practice, ACs are extensively used. This underlines the need for a proper AC scheme that relies on elementary, well-studied, cryptographic primitives to provide authenticity and confidentiality. First, we discuss some examples that illustrate the need for a scheme that provides both confidentiality and authenticity. Then, we give a general definition of an AC scheme and use it as a reference throughout this paper.
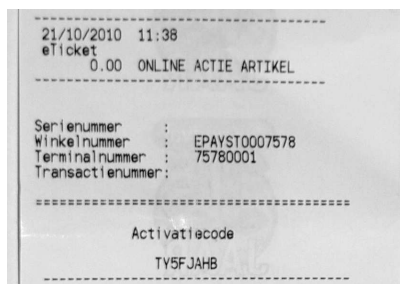

Fig. 1: Activation Code

Our Contribution This paper addresses some known methods that are used to generate ACs and proposes BEPAC, an AC scheme that combines best effort with best practice. BEPAC is based on well-studied cryptographic primitives to guarantee unique and authentic codes that provide a satisfactory level of confidentiality. Confidentiality is obtained by a Feistel construction. The Feistel construction has weak theoretical bounds when it is used on a small domain, therefore we do not rely on it for authenticity. We use the work of Black and Rogaway [3] on small domain encryption and make some small changes to achieve confidentiality. A practical attack on a general 3-round Feistel network is presented to demonstrate the weak bounds of the Feistel construction. For authenticity, we solely rely on a keyed-hash message authentication code (HMAC) of the serial number, where the size of this HMAC determines the probability of successfully guessing a valid AC. This separated design allows a separate analysis of both confidentiality and authenticity. An advantage of this approach is that authenticity is not automatically compromised when confidentiality is broken. Finally, a BEPAC solution fits on a smart card and therefore allows AC generation and clearance to be performed in a controlled environment.

Related Work Black and Rogaway [3] propose the Generalized Feistel Cipher (GFC) as a solution to small domain encryption. This elegant solution can be used to construct a permutation on any finite domain. In the BEPAC scheme we use their method in a slightly adapted way and solely to provide confidentiality. Black and Rogaway provide an adapted proof of Luby [15] to prove secrecy of a 3-round Feistel network. However, in their example configuration, the single DES round function does not give the 3×56-bit security since single DES

can be broken by exhaustive key search [25]. Moreover, in this setting it can be broken round by round which actually means that we have 58-bit security. Bellare et al. [2] propose 128-bit AES as a pseudo-random function which drastically increases the effort needed to break one round of the Feistel network by brute-force. However, the adaptive chosen-plaintext attack (CPA-2) on a 3-round Feistel network presented in this paper shows that the key length of the pseudo-random function used in each round does not have any influence on the attack complexity. Research on Feistel networks [22,20,19,15,13,21] has resulted in theoretical security bounds. Feistel constructions of six or more rounds are secure against adaptive chosen plaintext and chosen ciphertext attacks (CPCA-2) when the number of queries $m \ll 2^n$, see [22,13]. There is more related literature on the design of ACs, but to the best of our knowledge there are no proposals for the class of AC schemes that we discuss in this paper. Blundo et al. [4,5,8] introduce an e-coupon which is 420 bytes in size. This scheme uses a message authentication code (MAC) over some characterizing data like the identity of the manufacturer, name of the promoted product, expiry date etc. The resulting e-coupons contain valuable information but are too large to be entered manually by a user. In the work of Kumar et al. [14] and Jakobssen et al. [12] a coupon is basically a digital signature which also means that they describe relatively large e-coupons. Chang et al. [6] recognize the problem of efficiency and describe a scheme that is more suitable for mobile phones that have less processing power. On the one hand, they circumvent the use of public key cryptography which reduces the computational complexity, but on the other hand, their scheme describes relatively long codes. None of the schemes described previously satisfy the requirement of short codes that can be entered manually.

Matsuyama and Fujimura [17] describe a digital ticket management scheme that allows users to trade tickets. The authors discuss an account based and a smart card based approach and try to treat different ticket types that are solely electronically circulated. This in contrast with BEPAC which focuses on codes that can easily be printed on product wrappings. Our intention is not to define a trading system where ACs can be transfered from one person to another. Such contextual requirements are defined in RFC 3506 as Voucher Trading Systems (VTS). Terada et al. [28] come with a copy protection mechanism for a VTS and use public key cryptography. This makes the vouchers only suitable for electronic circulation. Furthermore, RFC 3506 and [17] do not discuss methods on how to generate these vouchers securely. We propose the BEPAC scheme in order to fill this gap.

2 AC Scheme Selection

This section first discusses some examples of AC systems. Then, two different approaches to set up an AC scheme are discussed and their main drawbacks are visited. After this, the Generalized Feistel Cipher of [3] is introduced in Section 2.1 which has some useful concepts that we use in our scheme. The focus of an AC scheme design lays on scalability, cost-efficiency and off-line use.

Finally, forgery of ACs should be hard, an adversary is only able to forge ACs with a very small predefined probability.

Examples of Activation Codes First, we discuss some examples of ACs in real life. A good first example is the *scratch prepaid card* that is used in the telecommunication industry. To use a prepaid card, the customer needs to remove some foil and reveals a code that can be used to obtain mobile phone credits. Then we have the *e-coupon* which is a widely used replacement for the conventional paper coupon. An e-coupon represents value and is used to give financial discount or rebate at the checkout of a web shop. The last example is a *one-time password* that gives access to on-line content. This content should only be accessible to authenticated people who possess this unique password; think of sneak previews of new material or software distribution.

All the aforementioned examples use unique codes that should be easy to handle, that is, people should be able to manually copy ACs without much effort. At the same time, it should be impossible for an adversary to use an AC more than once or autonomously generate a new valid AC. Altogether, ACs are unique codes that have to guarantee authenticity. An AC system provides authenticity when an adversary is not able to forge ACs. It is a misconception that authenticity only is enough for an AC scheme. In the end, most AC systems are used in a competitive environment. When vendor A starts a campaign where ACs are used to promote a product and provide it for free to customers, then it is not desirable that vendor B finds out details about this campaign like the number of released ACs. Other sensitive details might be the value that different ACs represent or the expiry date of ACs. It is for this reason that we need confidentiality, which means that an adversary is not able to recognize patterns or extract any information from a released AC.

Many systems use codes that need to provide the properties discussed above. In this paper we refer to all these codes as ACs. By an AC scheme $\mathcal{S}$ we indicate a tuple $(\mathcal{A}, \mathcal{N}, \mathcal{P}, \lambda)$ where $\mathcal{A}$ is the size of the used alphabet, $\mathcal{N}$ is the number of desired ACs, $\mathcal{P}$ determines the probability $P = \frac{1}{\mathcal{P}}$ of an adversary guessing an AC and λ is the length of the ACs.

Database Approach The database approach is very straightforward and consists of a database that contains all the released ACs and their current status. The generation of new AC entries is done by a pseudo-random function. When a customer redeems an AC, its status is set to 'used'. An advantage of the database approach is that the randomness of the ACs is directly related to the randomness of the pseudo-random generator. So, it is important to select a good pseudo-random generator, e.g. a FIPS certified one. On the other hand, the protection of this valuable data is still a problem. For instance, if an attacker manages to add entries to the database or is able to change the record status to 'unused', it will be hard to detect this fraud in time. Also, it is necessary to check any new AC against all existing entries since there might be a collision. As a consequence, access to the complete set of ACs is needed on generation of new ACs.

Block Cipher Approach Another approach is to use a block cipher that gives a random permutation $F : \{0,1\}^n \to \{0,1\}^n$ from serial numbers to ACs. The provider maintains a counter i to keep track of the number of generated ACs. This way the authenticity of a serial number can be checked since only ACs that decrypt to a serial $< i$ are valid. A disadvantage of this method is the size of the resulting AC which is 128-bits for AES and 64-bits for 3DES. For AES, this results in a string of about 21 characters when we use numbers, upper- and lowercase characters in our alphabet. For 3DES, this is about 11 characters. Smaller block ciphers do exist, like KATAN [10] which is 32-bits, but are not very well-studied. Furthermore, block ciphers force ACs to have a length that is a multiple of the block size b. An alternative could be the concept of elastic block ciphers [9] which is an extended scheme where variable message sizes are allowed as input. Moreover, this scheme uses well-studied block ciphers. Still, the minimal size of a plaintext message is the block size b of the incorporated block cipher. So, this does not give any advantage and is still too large for our target, which is roughly 20 to 50-bit codes.

2.1 Small Domain Ciphers

Black and Rogaway introduce the Generalized Feistel Cipher (GFC) in [3]. The GFC is designed to allow the construction of arbitrary domain ciphers. Here, arbitrary domain means a domain space that is not necessarily $\{0,1\}^n$. For ACs we want to use a small domain cipher where the domain size can be customized to a certain extent, therefore we look into the proposed method in [3]. Before we describe the Generalized Feistel Cipher, we briefly visit the basic Feistel construction.

Feistel Network A Feistel network [16] is a permutation that takes an input x of size $2n$, then performs a number of rounds r with round functions $f_1, ..., f_r$, and finally delivers an output y of size $2n$. The input is split into two blocks $\langle L, R \rangle \in \{0,1\}^{2 \times n}$. As shown in Figure 2, every right block is input to a round function f_i. The output of this function is combined with the left block and becomes the new right block, e.g. $L' = f_1(R) + L$ for GFC. The original right block becomes the new left block. For the ease of decryption the last output blocks are swapped in case of an odd number of rounds (which is the case in Fig. 2).
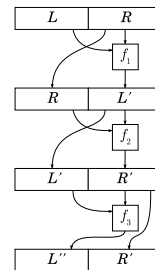

Fig. 2: Feistel

Generalized Feistel Cipher The GFC of Black and Rogaway [3] was introduced to handle flexible domain sizes. Take for example an encryption $E : 5^{14} \to 5^{14}$ which is not a domain that is captured by standard block cipher algorithms. The BEPAC scheme borrows some of the ideas of GFC to be able to construct arbitrarily sized AC configurations.

In GFC the left and right block of the Feistel network are "similarly sized" which means that their domain size may deviate a little. For the particular

case of ACs we have looser restrictions on the arbitrariness of our domain and we can increase the guessing probability $\mathcal{P}$ to influence the domain size. As a consequence, the system parameters of BEPAC can be chosen such that the left and right block are equally sized.

An obvious way to use the Feistel network is to create a pseudo-random permutation $F : \mathcal{K} \times \mathcal{M} \to \mathcal{M}$ where $\mathcal{K}$ and $\mathcal{M}$ are the key space and message space respectively. To generate ACs, we take as plaintext an index i and use the resulting ciphertext as AC α. In order to check α the provider keeps track of the last index i and considers a given α valid when $F^{-1}(\alpha) \leq i$. This construction guarantees:

1. Collision-freeness, since F is a permutation.
2. Valid serial numbers, they cannot be predicted since F is a pseudo-random permutation.

As Black and Rogaway already conclude in [3] the Generalized Feistel Cipher has weak security bounds when used in applications where the message space is roughly from $k = 2^{30}$ up to $k = 2^{60}$. This suggests that our second argument might not be that strong.

Also, the serial i is kept secret and one might argue that this presumes unforgeability. However, the way i is embedded allows an adversary to make useful assumptions about i since the ACs are generated using consecutive numbers. In the GFC, the left block L and right block R are initiated as follows:

$$L = i \bmod 2^n, \qquad R = \lfloor i/2^n \rfloor$$

Here, L represents the least significant bits of i, and the successor of every i always causes a change in L. On the contrary, when i is sequentially incremented, the value of R changes only once every 2^n times. This way, the first 2^n ACs are generated with $R = 0$, the second 2^n ACs with $R = 1$, etc. The problems that this little example already points out are further explained in the next section.

3 Feistel Permutation Recovery using CPA-2

In this section we present a practical attack on a three-round Feistel construction in order to illustrate the problem of choosing a small number of rounds and using a serial embedding as suggested by Black and Rogaway [3].

Theorem 1. *Consider a three-round 2n-bit Feistel construction. Then there exists an algorithm that needs at most 2^{n+2} adaptive chosen plain-/ciphertext pairs to compute any ciphertext from any plaintext and vice versa without knowledge of the secret round keys and regardless the used key length.*

Proof. The two ciphertext blocks are defined in terms of the plaintext blocks as follows:

$$\begin{aligned}
R' &= f_2(f_1(R) + L) + R \\
L'' &= f_3(f_2(f_1(R) + L) + R) + f_1(R) + L \\
&= f_3(R') + f_1(R) + L
\end{aligned} \tag{1}$$

Note that L'' uses R' as input to $f_3(\cdot)$. With L_i we denote $L = i$ and similarly R_j denotes $R = j$. The notation $R'_{(i,j)}$ means the value of R' when L_i and R_j are used as input blocks. We first observe that several triples (f_1, f_2, f_3) lead to the same permutation and show that it is always possible to find the triple with $f_1(0) = 0$. To this end, if we replace the triple (f_1, f_2, f_3) with the triple (f'_1, f'_2, f'_3) defined by Equation (2), this leads to the same permutation (Equation (1)) with the desired property that $f'_1(0) = 0$.

$$f'_1(x) = f_1(x) - f_1(0), \qquad f'_2(x) = f_2(x + f_1(0)), \qquad f'_3(x) = f_3(x) + f_1(0) \quad (2)$$

So, without loss of generality we may assume that $f_1(0) = 0$.

Next, we describe a method to find a triple (f_1, f_2, f_3) with $f_1(0) = 0$. First, we determine f_2, then f_1 and finally f_3.

By Equation (1) we get f_2:

$$f_2(f_1(R_0) + L_i) = R'_{(i,0)} - R_0 = R'_{(i,0)} \qquad \Rightarrow \qquad f_2(L_i) = R'_{(i,0)} \quad (3)$$

Now, to find f_1 observe that $f_1(j)$ is a solution for x in the equation $f_2(x) = R'_{(0,j)} - R_j$. However, this equation does not always have one unique solution since f_2 is a pseudo-random function. In case of multiple solutions we compare the output of successive (wrt. x) function inputs with the values for $f_2(x + i)$ that were found using Equation (3). Then, the correct x is the unique solution to:

$$f_2(x + i) = R'_{(i,j)} - R_j \qquad \text{for } i = 0, \ldots, m \quad (4)$$

Sometimes $m = 0$ already gives a unique solution. At the end of this section we show that with very high probability $m = 1$ defines a unique solution. We find:

$$f_1(j) = x \quad (5)$$

When f_1 and f_2 are both determined, L and R can be chosen such that every value for $R' \in \{0, \ldots, 2^n - 1\}$ is visited. Since R' functions as direct input to f_3 it is possible to find all input-output pairs for f_3. To visit every possible input value $z = 0, \ldots, 2^n - 1$ find a pair L_i, R_j such that $R'_{(i,j)} = z$. First, find an index x such that $f_2(x) = z - R_j$. If such an x does not exist choose a different value for R_j. There is always a solution for x since R_j covers the whole domain of f_2. Second, derive L_i:

$$f_2(f_1(R_j) + L_i) = f_2(x) \qquad \Rightarrow \qquad L_i = x - f_1(R_j) \quad (6)$$

Note that the determination of L_i and R_j does not need any intermediate queries since it is completely determined by f_1 and f_2. Next, we query the system with L_i and R_j and use Equation (3) and (5) to compute f_3 as follows:

$$f_3(x) = L''_{(i,j)} - f_1(R_j) - L_i \quad (7)$$

This completes the solution for a triple (f_1, f_2, f_3) that results in the same permutation as the Feistel construction under attack.

Number of Queries The determination of f_2 is given by Equation (1) and costs 2^n queries. The determination of f_1 is given by Equation (4). The probability p that there exists an $x' \neq x$ for a preselected x such that

$$(f_2(x) = f_2(x')) \ \wedge \ \left(\bigwedge_{i=1,\ldots,m} f_2(x + i) = f_2(x' + i) \right) \tag{8}$$

can be split into two parts. First, we have the probability p_1 that there is a collision $f_2(x) = f_2(x')$ with $x \neq x'$. Then, the second probability p_2 covers cases where a preselected position $f_2(x + i)$ has the same value as some other preselected position $f_2(x' + i)$. We take $k = 2^n$ and the two probabilities are then given by $p_1 = 1 - (\frac{k-1}{k})^{(k-1)}$ and $p_2 = \frac{1}{k}$. Now, $p = p_1 \cdot p_2{}^m$ because we need to multiply by p_2 for every other successive match. To conclude, the probability that there is an $x' \neq x$ for a Feistel construction of size $2 \cdot k$ and with m successive queries, i.e. the probability that there is no unique solution x to Equation (4), is the probability

$$p = \frac{1}{k^m} - \frac{1}{k^m} \cdot \left(\frac{k-1}{k} \right)^{k-1} \tag{9}$$

So, $p < \frac{1}{k^m}$ and depending on the size k, $m = 1$ already gives p close to zero. In practice one might sometimes need an additional query ($m = 2$) or only one query ($m = 0$), but on average $m = 1$. This means that the cost for determination of f_1 is $2 \cdot 2^n$ queries on average. Then, the determination of f_3 is given by Equation (7) and costs at most 2^n queries. As a result, the determination of (f_1, f_2, f_3) has an upper bound of 2^{n+2} queries.

4 BEPAC Scheme

In this section we propose our Activation Code Scheme called BEPAC. Its primary objective is to ensure authenticity and its secondary objective is to provide confidentiality. Confidentiality is satisfied up to the security bounds given by Black and Rogaway in [3]. In the BEPAC scheme, loss of confidentiality does not affect the authenticity property.

The authenticity is achieved in an obvious way by the use of an HMAC which is a keyed hash function. We take the truncated HMAC h of a sequence number i and concatenate it to i itself. For this concatenation we use an embedding m like the one used by Black and Rogaway in [3] and Spies in [2]. We rely on the strength of the underlying hash function which covers the *best practice* part of our solution: ACs are not forgeable. The length of an HMAC is usually too long for the ease of use that is demanded for ACs. Therefore we introduce the probability $P = \frac{1}{\mathcal{P}}$ that puts a lower bound on the success rate of guessing correct ACs. We use this parameter to limit the length of the codes, i.e. $\mathcal{P}$ determines the size of the HMAC. A lower success probability for an adversary is achieved by concatenating a bigger part of the HMAC and thus results in a longer AC.

Our solution differs from encryption schemes for small domains [3,18] in the sense that we make a clear separation between the part that provides authenticity and the part that provides confidentiality. The latter is added as an additional operation on the embedding m. We use a balanced Feistel construction as proposed in [3] to create the necessary confusion and diffusion. This separation between authenticity and confidentiality is really different from an approach where the sequence number i is directly fed into a Feistel construction and when it solely depends on this construction for its authenticity. The attack in Section 3 demonstrates that we cannot rely on a Feistel construction for authenticity when it is used on a small domain. These results form the basis of our design decision.

4.1 AC Scheme Setup

The BEPAC scheme setup is a construction (see Fig. 3) where an embedding m of an index i and a part of HMAC(i) are fed into a Feistel network. Since this is a balanced Feistel network, m needs to be divided into two equally sized blocks. When this is not possible a small part h' of HMAC(i) bypasses the Feistel network and is embedded together with the cryptogram c from the Feistel network to form AC α.

The BEPAC scheme $\mathcal{S}$ is a tuple $(\mathcal{A}, \mathcal{N}, \mathcal{P}, \lambda, \omega)$ where $\mathcal{A}$ is the size of the alphabet, $\lambda + \omega$ is the length of the ACs where λ is always even and ω is either 0 or 1. Then, $\mathcal{N}$ is the number of ACs and $\mathcal{P}$ determines the probability $P = \frac{1}{\mathcal{P}}$ of obtaining a valid AC by a random guess, e.g. $\mathcal{P} = 10.000$. We assume $\mathcal{A} < \mathcal{P}$.

Fig. 3: BEPAC Scheme

Definition 1 (Valid AC Scheme). *An AC scheme* $\mathcal{S} = (\mathcal{A}, \mathcal{N}, \mathcal{P}, \lambda, \omega)$ *is valid when* $\mathcal{A}^\lambda \geqslant \mathcal{N} \times \mathcal{P} \times \mathcal{A}^{-\omega}$ *holds and* λ *is even.*

A valid AC scheme $\mathcal{S}$ can be obtained as follows:

(a) The user chooses the alphabet size $\mathcal{A}$, desired number of ACs $\mathcal{N}$ and some minimal guess probability $\frac{1}{\mathcal{P}}$.
(b) Now the minimal length λ is calculated such that $\mathcal{A}^\lambda >= \mathcal{N} \times \mathcal{P}$ by taking $\lambda = \lceil {}^{\mathcal{A}}\!\log(\mathcal{N} \times \mathcal{P}) \rceil$
(c) $|\mathcal{A}^\lambda - \mathcal{N} \times \mathcal{P}|$ is minimized by taking $\mathcal{P} = \lfloor \mathcal{A}^\lambda / \mathcal{N} \rfloor$
(d) The length λ can be either odd or even:
   − When $\lambda = 2k+1$ and $\mathcal{A} < \mathcal{P}$ then we adjust $\mathcal{P}$ such that $\mathcal{A}$ is a divisor of $\mathcal{P}$. As a consequence, we might have a larger number of ACs $\mathcal{N}$.

$$\mathcal{P} = \mathcal{P} - (\mathcal{P} \bmod \mathcal{A}), \qquad \mathcal{N} = \lfloor \mathcal{A}^\lambda / \mathcal{P} \rfloor$$

   After these operations we obtain the system $\mathcal{S} = (\mathcal{A}, \mathcal{N}, \mathcal{P}, \lambda - 1, 1)$.
   − When $\lambda = 2k$ we obtain the system $\mathcal{S} = (\mathcal{A}, \mathcal{N}, \mathcal{P}, \lambda, 0)$.
(e) The process is repeated from step (a) when no valid system $\mathcal{S}$ is found.
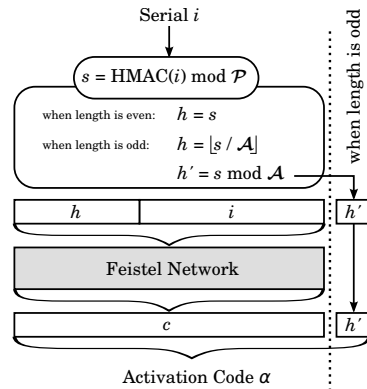
4.2 Generation

This section describes the generation of new ACs once a valid AC scheme is configured. Algorithm 1 contains the pseudo code for AC generation. The plaintext is an embedding m of a part of $\mathrm{HMAC}(i)$ and i itself. In case of an odd AC length ($\omega = 1$) a small part h' of $\mathrm{HMAC}(i)$ is excluded from this embedding. The part of $\mathrm{HMAC}(i)$ that is used in m is determined by $\mathcal{P}$.

$$s = \mathrm{HMAC}(i) \bmod \mathcal{P}, \qquad h = \lfloor s \times \mathcal{A}^{-\omega} \rfloor, \qquad h' = s \bmod \mathcal{A}, \qquad m = h \times \mathcal{N} + i$$

The balanced Feistel construction is defined with:

$$k = \mathcal{A}^{(\lambda/2)}, \qquad L = m \bmod k, \qquad R = \lfloor m \ / \ k \rfloor$$

L and R are input blocks with size k of a balanced Feistel network. We denote the output blocks after r rounds by $L^\star$ and $R^\star$. When the number of rounds r is even the cryptogram c is given by:

$$c = R^\star \times k + L^\star$$

When r is odd, the left and right block are swapped and the cryptogram c is given by:
$$c = L^\star \times k + R^\star$$

This difference between odd and even is there to allow the same construction for encoding and decoding. Finally, the activation code α is given by:

$$\alpha = c \times \mathcal{A}^\omega + \omega h'$$

4.3 Verification

This section describes the verification of previously generated ACs for a valid AC scheme. Algorithm 2 contains the pseudo code for AC verification. Given an AC α and an AC scheme $\mathcal{S}$ the validity can be checked as follows. First compute c and h' from α:
$$c = \lfloor \alpha/\mathcal{A}^\omega \rfloor, \qquad h' = \alpha \bmod \mathcal{A}^\omega$$
The balanced Feistel construction is defined with input block size $k = \mathcal{A}^{(\lambda/2)}$. Now, the input blocks L and R are obtained from c as follows:

$$L = c \bmod k, \qquad R = \lfloor c/k \rfloor$$

L and R are input blocks with size k of a balanced Feistel network. We denote the output blocks after r rounds by $L^\star$ and $R^\star$. In this case we want to decrypt and therefore use the round keys in reverse order. When the number of rounds r is even the plaintext m is given by:

$$m = R^\star \times k + L^\star$$

When r is odd, the left and right block are swapped and the plaintext m is given by:

$$m = L^\star \times k + R^\star$$

Now, we are able to obtain the partial HMAC h and index i from m by:

$$h = \lfloor m/\mathcal{N} \rfloor, \qquad i = m \bmod \mathcal{N}$$

We calculate the partial HMAC h_t and h'_t like in the encoding, but now we use the recovered index i. Finally, we say that α is a valid AC iff $h_t = h$ and $\omega h'_t = \omega h'$.

Algorithm 1 GENERATE(i)	**Algorithm 2** VERIFY(α)

$k \leftarrow \mathcal{A}^{(\lambda/2)}$
 $s \leftarrow \mathrm{HMAC}(i) \bmod \mathcal{P}$
 $h \leftarrow \lfloor s \times \mathcal{A}^{-\omega} \rfloor$
 $h' \leftarrow s \bmod \mathcal{A}$
 $m \leftarrow h \times \mathcal{N} + i$
 $L \leftarrow m \bmod k; \; R \leftarrow \lfloor m/k \rfloor$
 for $j \leftarrow 1$ **to** r **do**
    $tmp \leftarrow (L + f_j(R)) \bmod k$
    $L \leftarrow R; \; R \leftarrow tmp$
 end for
 if r is odd **then**
    $c \leftarrow L \times k + R$
 else
    $c \leftarrow R \times k + L$
 end if
 $\alpha \leftarrow c \times \mathcal{A}^\omega + \omega h'$
	$k \leftarrow \mathcal{A}^{(\lambda/2)}$
	 $c \leftarrow \lfloor \alpha/\mathcal{A}^\omega \rfloor; \; h' \leftarrow \alpha \bmod \mathcal{A}^\omega$
	 $L \leftarrow c \bmod k; \; R = \lfloor c/k \rfloor$
	 for $j \leftarrow r$ **to** 1 **do**
	    $tmp \leftarrow (L + f_j(R)) \bmod k$
	    $L \leftarrow R; \; R \leftarrow tmp$
	 end for
	 if r is odd **then**
	    $m \leftarrow L \times k + R$
	 else
	    $m \leftarrow R \times k + L$
	 end if
	 $h \leftarrow \lfloor m/\mathcal{N} \rfloor; \; i \leftarrow m \bmod \mathcal{N}$
	 $s \leftarrow \mathrm{HMAC}(i) \bmod \mathcal{P}$
	 $h_t \leftarrow \lfloor s \times \mathcal{A}^{-\omega} \rfloor$
	 $h'_t \leftarrow s \bmod \mathcal{A}$
	 if $h_t = h$ **and** $\omega h'_t = \omega h'$ **then**
	    VALID
	 else
	    INVALID
	 end if

5 Example Application: Smart Card

In this section we want to give an example of an Activation Code System (ACS). In an ACS there are a few things that need to be managed. The index i of the latest generated AC and the ACs that have been used so far. Since this information is highly valuable and represents financial value it must be well protected. Think of an application where the ACs are printed on prepaid cards covered by some scratch-off material. The production of these cards is a very secured and well-defined process to ensure that activation codes are kept secret during manufacturing. These cards need to have all kinds of physical properties, e.g. the AC should not be readable when the card is partly peeled off from the back. This can be achieved by printing a random pattern on top of the scratch-off foil.

At some point there is a very critical task to be executed when the ACs need to be delivered to the manufacturer. An obvious method to do this is to encrypt the list of ACs with a secret key. Later on in the process, this list of randomly generated codes needs to be maintained by the vendor who sells the scratch cards. This induces a big security threat since leakage of this list or

Desired		$\mathcal{A}=8$				$\mathcal{A}=20$				$\mathcal{A}=31$			
$\mathcal{N}$	$\mathcal{P}$	$\mathcal{N}$	$\mathcal{P}(\times 10^3)$	l	Bits	$\mathcal{N}$	$\mathcal{P}(\times 10^3)$	l	Bits	$\mathcal{N}$	$\mathcal{P}(\times 10^3)$	l	Bits
10^1	10^3	10^1	26,214	6	18	10^1	16	4	18	10^1	92,352	4	20
10^2	10^3	10^2	20,968	7	21	10^2	32	5	22	10^2	286,285	5	25
10^3	10^3	10^3	16,777	8	24	10^3	64	6	26	10^3	28,613	5	25
10^4	10^3	10.004	13,416	9	27	10^4	128	7	31	10^4	88,75	6	30
10^5	10^3	100.003	10,737	10	30	10^5	12,8	7	31	10^5	275,125	7	35
10^6	10^3	1.000.006	68,719	12	36	10^6	25,6	8	35	1.000.567	27,497	7	35
10^7	10^3	10.001.379	54,968	13	39	10^7	51,2	9	39	10.000.012	85,289	8	40
10^8	10^3	100.001.057	43,98	14	42	10^8	102,4	10	44	100.010.675	264,368	9	45
10^9	10^3	1.000.010.575	35,184	15	45	10^9	10,24	10	44	1.001.045.818	26,412	9	45

Table 1: BEPAC Configurations

unauthorized modification results in financial loss. Especially when it directly relates to the core business like in the telecommunications industry.

The use of a secure application module (SAM) significantly reduces this risk. A SAM is typically a tamper-resistant device, often a smart card, which is in most cases extensively tested and certified in accordance to a standard, e.g. the Common Criteria[3]. The elegance of the solution presented in this paper is that it can be implemented using smart cards. The supplier determines the probability P, number of codes $\mathcal{N}$, size of character set $\mathcal{A}$ and the key K to be used. An obvious approach is to use two smart cards since the production and clearance of activation codes are very likely to happen at two different locations. Both smart cards are initialized using the same AC scheme $\mathcal{S}$ and the same key K. From that moment on the first one only gives out up to $\mathcal{N}$ new activation codes. The second one is used at the clearance house to verify and keep track of traded activation codes. This can be done by a sequence of bits where the i-th bit determines whether the i-th activation code has been cleared. For 1.000.000 activation codes approximately 122 kB of storage is needed. This fits on a SmartMX card [27] which is available with 144 kB of EEPROM. Of course, multiple cards can be used if more ACs are needed.

6 Analysis

In this section we discuss the system parameters of BEPAC and decide on some minimal bounds and algorithms. We tested a 6-round BEPAC scheme for obvious flaws using the NIST random number test [26]. This test implementation also delivered the numbers in Table 1 which give a good indication of the length l of the codes compared to different AC scheme configurations. In the left column the desired values are given for the number of codes $\mathcal{N}$ and the guessing probability $\mathcal{P}$. We tested these different numbers for three different alphabet sizes $\mathcal{A}$.

Number of Rounds We found good arguments to set the minimum number of rounds to six for the BEPAC scheme. The literature shows that Feistel constructions of six or more rounds are secure against adaptive chosen plaintext and chosen ciphertext attacks (CPCA-2) when the number of queries $m \ll 2^n$,

---

[3] http://www.commoncriteriaportal.org

see [22,13]. Patarin [21] shows that an adversary needs at least $2^{3n/4}$ encryptions to distinguish a six-round Feistel construction from a random permutation. A six round Feistel network sufficiently covers the risk of leaking serial number information, but this is of course a minimum.

Key Derivation In the BEPAC scheme we need different round keys for every Feistel round and another different key for the calculation of the HMAC on the serial. We propose to derive these keys from an initial randomly chosen key [1] by a key derivation function (KDF). There are several definitions available for KDFs and we propose to use KDF1 which is defined in ISO 18033-2 [11]. Recommendations for KDFs and their construction can also be found in [7]. The first key that is derived is used as the key in the HMAC calculation of h and h' (Section 4). After this the round keys for the Feistel construction are successively derived.

Pseudo-random Functions Furthermore, we need to decide on the pseudo-random functions (PRFs) that are used as round functions of the Feistel network. The pseudo-randomness of the permutation defined by a Feistel network depends on the chosen PRF in each round [15]. It is straightforward to use a cryptographic hash function since we already need a hash function for the HMAC [24] calculation and it keeps our AC scheme simple.

Hash Function In the end, the BEPAC scheme is solely based on a single cryptographic hash function. We follow the secure hash standard FIPS 180-3 [23] and propose to use an approved hash function like SHA-256.

7   Conclusions

In this paper we have introduced activation codes (ACs), short codes of fixed length, that represent value. These ACs should be scalable, cost efficient and forgery resistant. In the literature, several solutions [4,5,8,14,12,6,17,28,17] handle digital coupons or tickets that are somehow reminiscent to our notion of ACs. The difference is that most solutions use public key cryptography or other means that result in lengthy codes. In fact, these solutions come closer to some extended notion of digital cash and are not meant to give a solution on the generation of ACs. To the best of our knowledge there is no scheme that focuses on the class of ACs that are described in this paper (roughly think of 20 to 50-bit codes). Our proposed AC scheme for this class satisfies authenticity and confidentiality in a way that when confidentiality is compromised it does not automatically break authenticity and vice versa.

In order to allow a relatively small and arbitrary message space for our AC scheme we use some of the ideas of Black and Rogaway [3] in their Generalized Feistel Cipher to satisfy the confidentiality in our scheme. Several studies [22,20,19,15,13,21] show that the security bounds of Feistel constructions are not strong enough and thus make the use of Feistel constructions in small domains questionable. To illustrate this, we have demonstrated that CPA-2 allows

an adversary to recover the complete permutation from only 2^{n+2} plaintext-ciphertext pairs. Still, the Feistel construction is suitable for the purpose of confidentiality in our AC scheme. Since confidentiality is a secondary goal, it relaxes the demands on the security bounds. Furthermore, in BEPAC the plaintext cannot be predicted which counters the attacks from the literature. And most important of all, a Feistel construction defines a permutation on the AC domain, which means in practice that we do not have to store any additional data in order to remember which ACs are already published and which are not.

To conclude, we found satisfactory system parameters for the minimum number of Feistel rounds and we defined a method to do the key derivation for the round keys. Furthermore, we suggested a specific pseudo-random function (PRF) and hash function for concrete implementations. Finally, we have implemented the BEPAC scheme[4] and performed some statistical tests using the NIST Random Number Test [26]. This test did not reveal any obvious flaws.

It might be interesting for future work to create a smart card implementation of BEPAC as suggested in Section 5.

References

1. E. Barker and J. Kelsey. Recommendation for Random Number Generation Using Deterministic Random Bit Generators (revised). *NIST Special publication*, 800:90, 2007.
2. M. Bellare, P. Rogaway, and T. Spies. Format-Preserving Feistel-Based Encryption Mode. `http://csrc.nist.gov/groups/ST/toolkit/BCM/modes_development.html`, November 2009.
3. J. Black and P. Rogaway. Ciphers with Arbitrary Finite Domains. *Topics in Cryptology–CT-RSA 2002*, pages 185–203, 2002.
4. C. Blundo, S. Cimato, and A. De Bonis. A Lightweight Protocol for the Generation and Distribution of Secure E-Coupons. In *WWW '02: Proceedings of the 11th international conference on World Wide Web*, pages 542–552, New York, NY, USA, 2002. ACM.
5. C. Blundo, S. Cimato, and A. De Bonis. Secure E-Coupons. *Electronic Commerce Research*, 5(1):117–139, 2005.
6. C.C. Chang, C.C. Wu, and I.C. Lin. A Secure E-coupon System for Mobile Users. *International Journal of Computer Science and Network Security*, 6(1):273, 2006.
7. L. Chen. Recommendation for Key Derivation Using Pseudorandom Functions. *NIST Special Publication*, 800:108, 2009.
8. S. Cimato and A. De Bonis. Online Advertising: Secure E-Coupons. *Theoretical Computer Science*, pages 370–383.
9. D. Cook, A. Keromytis, and M. Yung. Elastic Block Ciphers: The Basic Design. In *Proceedings of the 2nd ACM symposium on Information, computer and communications security*, pages 350–352. ACM, 2007.
10. C. De Cannière, O. Dunkelman, and M. Knežević. KATAN and KTANTAN – A Family of Small and Efficient Hardware-Oriented Block Ciphers. *Cryptographic Hardware and Embedded Systems-CHES 2009*, pages 272–288, 2009.

---

[4] `http://www.cs.ru.nl/~gkoningg/bepac`

11. ISO/IEC 18033-2:2006. *Information Technology - Security Techniques - Encryption Algorithms - Part 2: Asymmetric Ciphers*, 2006.

12. M. Jakobsson, P.D. MacKenzie, and J.P. Stern. Secure and Lightweight Advertising on the Web. *Computer Networks-the International Journal of Computer and Telecommunications Networkin*, 31(11):1101–1110, 1999.

13. L.R. Knudsen. The Security of Feistel Ciphers with Six Rounds or Less. *Journal of Cryptology*, 15(3):207–222, 2008.

14. M. Kumar, A. Rangachari, A. Jhingran, and R. Mohan. Sales Promotions on the Internet. *3rd USENIX Workshop on Electronic Commerce*, pages 167 – 176, 1998.

15. M. Luby. *Pseudorandomness and Cryptographic Applications*. Princeton University Press, 1996.

16. Michael Luby and Charles Rackoff. How to Construct Pseudorandom Permutations from Pseudorandom Functions. *SIAM J. Comput.*, 17(2):373–386, 1988.

17. K. Matsuyama and K. Fujimura. Distributed Digital-Ticket Management for Rights Trading System. In *Proceedings of the 1st ACM conference on Electronic commerce*, page 118. ACM, 1999.

18. B. Morris, P. Rogaway, and T. Stegers. How to Encipher Messages on a Small Domain. In *Proceedings of the 29th Annual International Cryptology Conference on Advances in Cryptology*, page 302. Springer, 2009.

19. M. Naor and O. Reingold. On the Construction of Pseudorandom Permutations: Luby-Rackoff Revisited. *Journal of Cryptology*, 12(1):29–66, 1999.

20. J. Patarin. New Results on Pseudorandom Permutation Generators Based on the DES Scheme. In *Advances in Cryptology-CRYPTO*, volume 91, pages 301–312.

21. J. Patarin. About Feistel Schemes with Six (or More) Rounds. *Lecture Notes in Computer Science*, 1372:103–121, 1998.

22. J. Patarin. Security of Random Feistel Schemes with 5 or More Rounds. *Lecture Notes in Computer Science*, pages 106–122, 2004.

23. FIB PUB. FIPS 180-3: Secure HASH Standard (SHS). *Federal Information Processing Standards Publication*, 2008.

24. N.F. PUB. FIPS 198: Keyed-Hash Message Authentication Code (HMAC). *Federal Information Processing Standards Publication*, 198.

25. J.J. Quisquater and F.X. Standaert. Exhaustive Key Search of the DES: Updates and Refinements. *Special-Purpose Hardware for Attacking Cryptographic Systems*, 2005.

26. A. Rukhin, J. Soto, J. Nechvatal, M. Smid, E. Barker, S. Leigh, M. Levenson, M. Vangel, D. Banks, A. Heckert, et al. *A Statistical Test Suite for Random and Pseudorandom Number Generators for Cryptographic Applications*. NIST, 2000.

27. NXP Semiconductors. *P5Cx012/02x/40/73/80/144 Family: Secure Dual Interface and Contact PKI Smart Card Controller*. NXP, June 2010.

28. M. Terada, M. Hanadate, and K. Fujimura. Copy Prevention Scheme for Rights Trading Infrastructure. In *Smart Card Research and Advanced Applications: IFIP TC8/WG8*, page 51. Springer, 2000.