Codes Identifying Bad Signatures in Batches

Jarosław Pastuszak[1], Josef Pieprzyk[2], and Jennifer Seberry[2]

[1] Systems Research Institute
Polish Academy of Sciences
Warsaw, POLAND
jarek.pastuszak@bsb.com.pl
[2] Centre for Computer Security Research
School of IT and Computer Science
University of Wollongong
Wollongong, NSW 2522, AUSTRALIA
Josef_Pieprzyk@uow.edu.au
Jennifer_Seberry@uow.edu.au

Abstract. The work is concerned with identification of bad signatures in a sequence which is validated using batching. Identification codes (id-codes) are defined and their general properties are investigated. A taxonomy of id-codes is given. The generic construction for a wide range of id-codes is given and its instantiation using mutually orthogonal Latin squares is described. Hierarchical identification is studied for two cases when the identification procedure uses a family of id-codes and when there is a single underlying id-code. Remarks about future research conclude the work.

Keywords: Digital Signatures, Batch Verification, Identification Codes.

1 Introduction

The concept of digital signatures have evolved considerably over the last two decades. Handwritten signatures are not suitable in electronic environment especially in the context of the ease in which electronic documents can be copied and/or cut-and-paste manipulated. Digital signature were invented to prevent documents from illegal modification. Digital signature encapsulates both the contents of the document and the signer secret key in such a way that verification is public - every body who knows the corresponding public key of the signer and the document, can validate the signature. Digital money is a specific kind of signature which is signed (typically, blindly) by the bank and used by customers in (almost) the same way as the traditional cash with added important functionality – it can be used in transactions performed over the Internet (electronic commerce).

The reliance of e-Commerce on digital money has a dramatic impact on the computing load imposed on the bank. The bank has become the focal point

where all electronic money (digital signatures) are flowing. Observe that before the transaction is approved, the electronic money must be validated. Batch verification is an attractive short cut for signature validation saving time and computing resources. It is applicable whenever the verifier gets a large number of digital signatures generated by the same signer provided the signature exhibits the homomorphic property allowing signatures to be validated in batches in the expense of a single exponentiation.

If the batch passes the validation, all signatures are considered correct and are accepted. If however, the batch fails to pass the validation test, the verifier must identify invalid signatures in the batch. Clearly, rejection of the whole batch is not an option. A natural question arises: how to identify invalid signatures in the batch so the valid signatures can be accepted ? Additionally, one would expect that the identification process should be as efficient as possible.

2 Background

Batch verification makes sense if the signatures in a batch are related or generated by the same signer. There are two types of signatures which can be batched: RSA signatures and DSA (DSS) signatures. The RSA signatures use the fixed exponent (public key of the signer) for verification. Assume that we have n messages and their signatures. The signatures can be verified independently at the expense of n exponentiations. The batch containing all signature can be verified at the expense of a single exponentiation plus $(n-1)$ modular multiplications.

DSA signatures are based on exponentiation when the base is fixed and publicly known (the modular arithmetics is chosen by the signer). Again n signatures can be verified one by one at the expense of n exponentiations. The batch of n signatures are validated using a single exponentiation and $(n-1)$ modular additions.

These two methods of batch verification are insecure as an enemy who may know the verification process, may try to get the verifier to accept invalid signatures. The simplest method of attack would be to produce a forged signature and insert two copies of it in the batch in such a way that they cancel each other when the verification is performed, For instance in the RSA case, any batch which contains two forged pairs: (m_f, s_f) and (m_f^{-1}, s_f^{-1}) where m_f is the forged message (document) and s_f is the forged signature, passes the verification test.

Bellare, Garay and Rabin [1] developed verification tests which are secure against any attacker. The security of the test is measured by the probability that a contaminated batch passes it making the verifier to accept all invalid or bad signatures contained in the batch. The probability of slipping bad signatures through the test can be traded with efficiency.

The problem we address in this work is an efficient identification of bad signatures after the test fails. There is a general method of bad signature identification which is called "cut and choose" in [3] or "divide and conquer" in [4]. It takes a contaminated batch and splits it repeatedly until all bad signatures are identi-

fied. The efficiency of this method depends on the degree of contamination (or how many bad signatures are in the batch) and also on how the bad signatures are distributed in the batch.

Note that identification of bad signatures resembles the problem of error correction. To be able to correct errors, the code must clearly identify all positions on which errors have occurred. As observed in [4], error correcting codes can be applicable for bad signature identification. There is a major difference between error correcting codes and *identification codes* or *id-codes* which allow to identify bad signatures. Computations in error correcting codes are done in the binary field with EXCLUSIVE-OR addition (XOR). The interaction among valid and invalid signatures within the batch are governed by INCLUSIVE-OR (logical OR).

The work is structured as follows. The model for id-codes is studied in Section 3. Section 4 investigates general properties of id-codes. Section 5 discusses taxonomy of id-codes. The general construction based on OR-checking matrix and its instantiation based on mutually orthogonal Latin squares are given in Section 6. Hierarchical identification is described in Section 7. A discussion about further work on id-codes closes the work.

3 The Model

The problem we are dealing with is bad signature identification in a batch which has failed to pass the test. The test $\mathcal{T}$ is a probabilistic algorithm which takes a batch of an arbitrary length and produces a binary outcome accept/reject. It satisfies the following two general conditions:

1. Any clean batch (which contains all valid signatures) always passes the test.
2. A dirty batch (which contains one or more bad signatures) fails the test with an overwhelming probability. In fact, it is reasonable to assume that a dirty batch always fails the test.

We further suppose that the cost of running the test does not depend on the size of the batch. This assumption seems to be true for relatively small batches where the computation effort is equivalent to a fixed sequence of exponentiations (see [1]).

Definition 1. *Given a batch* $\mathcal{B}^u = \{(m_i, s_i) | i = 1, \ldots, u\}$ *of signed documents* (m_i *is the i-th document and* s_i *its signature). The identification code* $IC(u, t)$ *able to identify up to* t *bad signatures is a collection of sub-batches* $(\mathcal{B}_1, \ldots, \mathcal{B}_v)$ *where* $\mathcal{B}_i \subset \mathcal{B}^u$ *such that for any possible pattern of up to* t *bad signatures, the outcomes (the syndrome)*

$$S = (\mathcal{T}(\mathcal{B}_1), \ldots, \mathcal{T}(\mathcal{B}_v))$$

uniquely identifies all bad signatures.

The identification code $IC(u,t)$ can be equivalently represented by its $v \times u$ test-checking matrix $A = [a_{ij}]$ such that

$$a_{ij} = \begin{cases} 1 & \text{if } (m_i, s_i) \in \mathcal{B}_j \\ 0 & \text{otherwise} \end{cases}$$

Clearly, for a fixed size u of the batch, one would like to obtain a code $IC(u,t)$ with the parameter v as small as possible. Note that v indicates how many tests $\mathcal{T}$ must be run to identify all bad signatures and it can be considered as the parameter characterising the efficiency of the code. The parameter v is upper bounded by u as it is always possible to design a trivial code whose matrix A is the $u \times u$ identity matrix. This code is equivalent to serial validation of signatures one by one.

The following notation is introduced. The code $IC(u,t)$ is uniquely identified by its $(v \times u)$ test-checking matrix A. The entries of A are binary. Columns of the matrix A are indexed by u signatures in a batch. So the matrix A can be seen as a sequence of columns of the form

$$A = (A_1, \ldots, A_u)$$

The index of the i-th signature in the batch $\mathcal{B}^u$ is the i-th column A_i. A row specifies the corresponding sub-batch which includes all signatures for which the entries are 1.

Note that if the i-th signature is bad the syndrome produced for a batch contaminated by it is equal to A_i or $S(i) = A_i$. Given a batch $\mathcal{B}^u$ with t bad signatures. Assume further that the bad signatures have occurred on positions $(b_1, \ldots, b_t)$ in the batch $\mathcal{B}^u$. Their corresponding indices are $(A_{b_1}, \ldots, A_{b_t})$. Denote the syndrome produced for the batch as

$$S(b_1, \ldots, b_t) = A_{b_1} \vee \ldots \vee A_{b_t}$$

where $\vee$ is bit-by-bit inclusive (logical) OR. For example, if

$$A_1 = \begin{bmatrix} 1 \\ 1 \\ 0 \\ 0 \end{bmatrix} \text{ and } A_2 = \begin{bmatrix} 1 \\ 0 \\ 1 \\ 0 \end{bmatrix} \text{ then } A_1 \vee A_2 = \begin{bmatrix} 1 \\ 1 \\ 1 \\ 0 \end{bmatrix}.$$

4 Properties of Id-Codes

Using an information-theoretic arguments, we argue that there is a lower bound on the v parameter.

Theorem 1. *Given an id-code $IC(u,t)$ which always identifies correctly any t bad signatures in the batch of the size u. Then the number of tests (and the number of collections) v satisfies the following inequality*

$$v \geq \log_2 \sum_{i=0}^{t} \binom{u}{i} \tag{1}$$

Proof. Given a batch $\mathcal{B}^u$ of u elements contaminated by at most t bad signatures. The identification of bad signatures is possible if the syndromes are distinct for all patterns of i bad signatures ($i \leq t$) so knowing the syndrome, it is possible to determine the positions of bad signatures in the batch. Note that there are

$$\sum_{i=0}^{t} \binom{u}{i}$$

different identifiable patterns (including the pattern with no bad signature). Now if we have v sub-batches $(\mathcal{B}_1, \ldots, \mathcal{B}_v)$, then the test $\mathcal{T}$ applied for a single sub-batch $\mathcal{B}_j$; $0 \leq j \leq t$, provides a binary outcome (pass/fail) so the number of possible syndromes is 2^v. Clearly

$$2^v \geq \sum_{i=0}^{t} \binom{u}{i}$$

and the bound described by Equation (1) holds.

Obviously, searching for id-codes makes sense if they are better (take less tests) than the naive id-code which tests batches containing single signatures. From Theorem 1 we can derive an interesting corollary.

Corollary 1. *Id-codes better than the naive id-code exist only if* $t < u/2$.

Proof. Note that for $t \geq n/2$, the number of tests

$$2^v \geq \sum_{i=0}^{t} \binom{u}{i} \geq \sum_{i=0}^{u/2} \binom{u}{i} \geq \frac{1}{2} \sum_{i=0}^{u} \binom{u}{i} = 2^{u-1}$$

Thus the number of tests v must be at least $u - 1$ which is almost the same as for the naive id-code which requires u tests.

Definition 2. *An index* A_i *includes* A_j *if* $A_i \vee A_j = A_i$.

Given the matrix A of an id-code. Observe that if there are two columns $i \neq j$ such that the index A_i includes A_j, then the code is unable to identify whether there are a single bad signature with the syndrome A_i or two bad signatures with the syndrome $A_i \vee A_j$. In other words, the matrix A with such indices is not able to identify bad signatures with indices A_j and A_i. We say that the two indices *collide*.

Lemma 1. *Given identification coding with a* $(v \times u)$ *test-checking matrix* A. *Assume further that there is an index* A_i *(column* A_i*) such that its Hamming weight* $wt(A_i) = r$, *then the number of colliding indices with* A_i *is*

$$C_{\#}(A_i) = 2^r + 2^{v-r} - 2.$$

Proof. There are two cases where collision may occur

– the index A_i includes other indices ($A_i \vee A_k = A_i$) for some k,
– the index A_i is included in other indices ($A_i \vee A_k = A_k$).

For a given index A_i with its Hamming weight r, we can create $2^r - 1$ indices which are included in A_i – the first case. We can also create $2^{v-r} - 1$ indices which include A_i – the second case. In effect, we have to exclude $2^r + 2^{v-r} - 2$ indices.

Corollary 2. *To increase effectiveness of identification codes we should select weights of indices so the number of colliding indices is minimal. The smallest number of colliding indices occurs when the Hamming weight of all indices is* $\frac{v}{2}$.

Assume that we have two indices A_i and A_j. We can define the intersection of the two as $A_i \wedge A_j$ where $\wedge$ is bit-by-bit logical AND.

Lemma 2. *Given two indices* A_i *and* A_j *such that* $wt(A_i) = r_1$ *and* $wt(A_j) = r_2$. *Denote* $A_c = A_i \wedge A_j$ *– the maximal index which is contained in both* A_i *and* A_j *and* $wt(A_c) = r$. *Then the number of indices which collide with the pair* (A_i, A_j) *is*

$$C_{\#}(A_i, A_j) = 2^{v-r_1} + 2^{v-r_2} + 2^{r_1+r_2-r} - 2^{v+r-r_1-r_2} - 2^{r_1-r} - 2^{r_2-r}.$$

Proof. Denote $\mathcal{A} = \{A_1, \ldots, A_u\}$. Note that $C_{\#}(A_i, A_j) \geq C_{\#}(A_i \vee A_j)$ and becomes the equality only if $r = 0$. From Lemma 1, we can write

$$C_{\#}(A_i \vee A_j) = 2^{r_1+r_2-r} + 2^{v+r-r_1-r_2} - 2.$$

Denote $\#A_i$ and $\#A_j$ the numbers of colliding indices from A_i and A_j, respectively, which have not been considered among the indices from $A_i \vee A_j$. Thus, we have

$$C_{\#}(A_i, A_j) = C_{\#}(A_i \vee A_j) + \#A_i + \#A_j.$$

There are the following cases, the index

– collides with A_i – there are 2^{r_1} such indices,
– collides with $A_j \setminus A_i$ – there are 2^{r_2-r} such indices,
– collides with $\mathcal{A} \setminus (A_i \wedge A_j)$ – there are $2^{v+r-r_1-r_2}$ such indices.

Observe that indices colliding with $A_j \setminus A_i$ have been already counted in $C_{\#}(A_i \vee A_j)$. Further on, note that the zero index (all bits are zero) has been counted. Therefore

$$\#A_i = (2^{r_2-r} - 1)(2^{v+r-r_1-r_2} - 1) \text{ and } \#A_j = (2^{r_1-r} - 1)(2^{v+r-r_1-r_2} - 1).$$

Adding the numbers we obtain the final result.

Lemma 3. *Given identification code determined by its* $(v \times u)$ *matrix* A. *If there is a parameter* $k \leq u$ *and a sequence of indices* $(A_{i_1}, \ldots, A_{i_k})$ *such that*

$$\bigvee_{j=1}^{k} A_{i_j} = \begin{bmatrix} 1 \\ \vdots \\ 1 \end{bmatrix} \stackrel{def}{=} \mathbf{1}_v,$$

then the id-code can identify no more than k bad signatures. Where $\bigvee_{j=1}^{k}$ *stands for bit-by-bit logical OR and* $\mathbf{1}_v$ *is a binary vector of length v containing ones only.*

Proof. Denote $\mathcal{A} = \{A_1, \ldots, A_u\}$ as the set of all indices (columns) of the matrix A. Create the following two sets:

$$\mathcal{A}_1 = \{A_{i_1}, \ldots, A_{i_k}\} \text{ and } \mathcal{A}_2 = \mathcal{A} \setminus \mathcal{A}_1.$$

The proof proceeds by contradiction. Assume that any $t = k + 1$ bad signatures can be identified. Now we take a sequence of k bad signatures with their indices $(A_{i_1}, \ldots, A_{i_k})$. Their syndrome is $\mathbf{1}_v$. Now if there is an extra bad signature than the collection of t bad signatures have the same syndrome – there is a collision and we have obtained the contradiction.

Observe that while designing id-codes, one would need to avoid using two indices A_i, A_j such that $A_i = \neg A_j$ where $\neg$ is bit-by-bit negation as such id-code identifies at most two bad signatures.

5 Taxonomy of Id-Codes

From an efficiency point of view, the batch size is preferred to be as large as possible. This also means that the size of the batch determines the block size of the id-code. So to identify bad signatures efficiently, one would need a family of id-codes rather than a single id-code working for a batch of fixed size. On the other hand, there is a boundary on the block size of a id-code which typically reflects restrictions imposed on computing resources.

Given a batch $\mathcal{B}^n$ and an id-code $IC(u, t)$ There are two general classes of bad signature identification:

- flat identification – there is an id-code whose block size equals the size of the batch ($n = u$),
- hierarchical identification – the number of signatures u in the batch is bigger than the block size u ($n > u$).

Clearly, flat identification applies an id-code and if the number of bad signatures is smaller than t, it always works. Its natural extension for larger batch sizes, could be the division of the batch into sub-batches each of size u. Hierarchical identification applies merges signatures into sub-batches and treats them as single signatures so we get a sequence of u sub-batches. The code is applied to it and identifies up to t contaminated sub-batches. These sub-batches can be either subject to flat or again to hierarchical identification.

Assume that the efficiency of identification is measured by the number of tests $\mathcal{T}$ necessary to identify all bad signatures. Note that this measurement is equivalent to the number of rows in the matrix A which defines the id-code. Intuitively, the more bad signatures are in a batch, the more expensive the identification process is. Id-code can be categorised into:

– codes with constant workload – no matter what is the degree of contamination, the number of tests is constant and the code either succeeds (if the identification capability exceeds the degree of contamination) or fails,
– codes with contamination-dependent workload – the number of tests depends on the contamination. Again codes fail if the number of bad signatures exceeds their identification capabilities.

From a practical point of view, codes with contamination-dependent workload are very attractive as they trade efficiency with identification capability. The identification process starts by performing a limited number of tests allowing to identify a single bad signature. If this fails, the identification proceeds by trying a new tests which together with the old tests permit to identify two bad signatures. The process continues until all bad signatures are identified or the identification fails. Important feature of id-codes seems to be re-usability of previous tests.

The batch validation applies a specific id-code, say $IC(u,t)$. If the capability of the code (expressed by t) is smaller than the degree of batch contamination ℓ (ℓ is the number of bad signatures in the batch), then the failure is unavoidable. Consequently, the parameter t must be increased. Additionally, the work done so far is likely to be lost. Thus it is imperative, to make a "good" guess about the maximum degree of contamination (the parameter t). Clearly, statistical information gathered from the past can suggest such a guess. Note that the situation simplifies somewhat if the codes in hand trade efficiency with identification capability as the guess can be more pessimistic.

6 Constructions of Id-Codes

As we know, one would wish to have an identification code which allows for gradual increment of t with a possible re-use of all tests conducted for smaller ts. Now we present our main construction.

Definition 3. *A $(k+1)n \times n^2$ matrix A with binary elements is a* OR-checking matrix *if there are $k+1$ ones per column, n ones per row, and the inner product of any pair of columns is either zero or one.*

Lemma 4. *Given a $(k+1)n \times n^2$ OR-checking matrix A. Then the OR of any subset of k columns is unique for $k = 1, \ldots, n-1$.*

Proof. For convenience in typesetting we will write these columns as rows by transposing the matrix – so we are going to consider A^T. We consider any k rows but permute them so that the ones are moved to the left of each row as far as possible. We now consider a simple counting argument to look at the intersection patters of the rows. If any two rows have an intersection +1, the ones (written as x) will use a total of $\frac{1}{2}(k+1)(k+2)-1$ columns and be able to be represented as:

```
   k+1            k         k-1      ...   | 2
x x x ... x  0 0 ... 0  0 ... 0    ...   |00
```

```
x 0 0 ... 0  x x ... x  0 ... 0   ...  |00
0 x 0 ... 0  x 0 ... 0  x ... x   ...  |00
                    .
                    .
                    .
0 0 0 ... x  0 0 ... x  0 ... x   ...  |xx
```

If any pair of rows do not have intersection $+1$ then more than $\frac{1}{2}(k+1)(k+2)-1$ columns will be needed to represent the patters of ones but the last row will always have at least 2 elements $+1$ at the right of the row which have no element in the column above either of them which is non-zero.

Now suppose that the matrix yielded that any $k-1$ rows corresponding to bad signatures gave a unique OR but that there are two solutions which give the same result for k rows indicating bad signatures. We rearrange the rows in our pattern representative, if necessary, so one of these two solutions is the last row. We now consider the other solution. For the first $k-1$ vectors and the second solution to cover the same number of columns the second solution must have two $+1$ at the right of the row which have no element in the column above either of them non-zero. But this means the first and second solution have at intersection at least 2 ones contradicting the definition of the OR-checking matrix. Hence any collection of k rows produces OR sums which are distinct.

We note that this proof does not extend to a collection of $k+1$ rows because in that case we could only assume the last row to have more than one elements $+1$ at the right of the last row which has no element in the column above it which is non-zero. This does not lead to any contradiction.

Corollary 3. *Given a $(k+1)n \times n^2$ OR-checking matrix A whose every two column intersection is either zero or one. Then there is an $IC(u,t)$ code which is capable to identify up to $t = n-1$ bad signatures within a batch of size $u = n^2$.*

The identification code based on OR-checking matrices is efficient as it allows to re-use all previous results if the guess about the parameter t has been wrong. Given a batch $\mathcal{B}^u$ of the size $u = n^2$. The $(n \times u)$ OR-checking matrix A is created. Denote $A^{(t)}$ as a shortened version of A containing first $(t+1)n$ rows of A; $t = 1, 2, \ldots, n-1$.

1. The identification process starts from the assumption that $t = 1$. First collection of $2n$ tests $\mathcal{T}$ are run for batches defined by rows of the matrix $A^{(1)}$. If the bad signatures are not correctly identified (i.e. the batch without bad signatures still fails the test $\mathcal{T}$), then it is assumed that $t = 2$. Otherwise the process ends.
2. Assume that the identification using $A^{(t)}$ has failed to identify bad signatures $(t = 2, 3, \ldots, n-1)$. The collection of necessary tests are defined by $A^{(t+1)}$. Note that $A^{(t+1)}$ differs from $A^{(t)}$ in that it contains n additional rows. The identification process can be accomplished by running n additional tests corresponding to the batches defined by rows in $A^{(t+1)}$ which are not in $A^{(t)}$. If the identification has not been successful, t is increment by 1 and the process continues.

The identification fails if $t \geq n$.

The construction also gives the upper bound on the number v of necessary tests to identify t bad signatures.

Corollary 4. *The number v of tests necessary to identify t bad signatures in the batch of size u satisfies the following inequality:*

$$v \leq (t+1)\sqrt{u}$$

There are many combinatorial structures which can be used to give the required OR-checking matrices for example transversal designs and group divisible designs. However we give a rich construction based on Latin squares.

A *Latin square* of order n is an $n \times n$ array in which n different symbols, say $a, b, \ldots$ each occur once in each row and column. Two Latin squares are said to the *mutually orthogonal* if when the squares are compared element by element each of the distinct pairs occurs exactly once. Formally, two Latin squares, L and L' are said to be mutually orthogonal if $L(a,b) = L(c,d)$ and $L'(a,b) = L'(c,d)$, implies $a = c$ and $b = d$. For further information, refer to [2].

Lemma 5. *Suppose there are k mutually orthogonal Latin squares of order n. Then there is a $(k+1)n \times n^2$ OR-checking matrix.*

Proof. We use the auxiliary matrices described in [2].

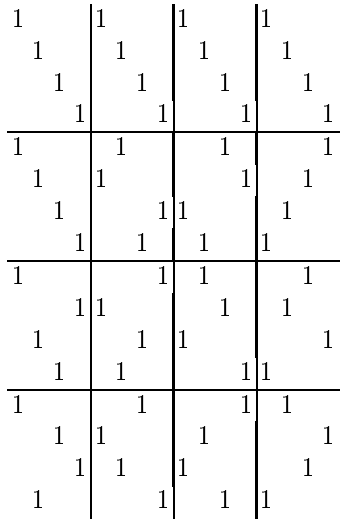
Example 1. Let

$$M_1 = \begin{bmatrix} \mathrm{a} & \mathrm{b} & \mathrm{c} & \mathrm{d} \\ \mathrm{b} & \mathrm{a} & \mathrm{d} & \mathrm{c} \\ \mathrm{c} & \mathrm{d} & \mathrm{a} & \mathrm{b} \\ \mathrm{d} & \mathrm{c} & \mathrm{b} & \mathrm{a} \end{bmatrix}, M_2 = \begin{bmatrix} \mathrm{a} & \mathrm{b} & \mathrm{c} & \mathrm{d} \\ \mathrm{c} & \mathrm{d} & \mathrm{a} & \mathrm{b} \\ \mathrm{d} & \mathrm{c} & \mathrm{b} & \mathrm{a} \\ \mathrm{b} & \mathrm{a} & \mathrm{d} & \mathrm{c} \end{bmatrix}, M_3 = \begin{bmatrix} \mathrm{a} & \mathrm{b} & \mathrm{c} & \mathrm{d} \\ \mathrm{d} & \mathrm{c} & \mathrm{b} & \mathrm{a} \\ \mathrm{b} & \mathrm{a} & \mathrm{d} & \mathrm{c} \\ \mathrm{c} & \mathrm{d} & \mathrm{a} & \mathrm{b} \end{bmatrix}$$

be three mutually orthogonal Latin squares of order 4 on the symbols $x_1 = a$, $x_2 = b$, $x_3 = c$ and $x_4 = d$. Define $M_{ij},\ 1 \leq i \leq k$, by

$$(M_{ij})_{ef} = \begin{cases} 1 & (M_i)_{fj} = x_e, \\ 0 & \text{otherwise.} \end{cases}$$

where $1 \leq e, f \leq 4$. So $M_{ij},\ 1 \leq i \leq 4$ and $1 \leq j \leq 4$ can be written as



Corollary 5. *Let $q > 2$ be a prime power then there are $q - 1$ mutually orthogonal Latin squares of order q*

Many other results are also known, for example for every $n \geq 3$ except 6 there are at least two orthogonal Latin squares of order n and for $n > 90$ there are at least 6.

7 Hierarchical Identification

Identification codes are designed to work with a batch of fixed size. In practice, one would expect to have an identification scheme which is going to work with a batch of arbitrary length. Hierarchical identification provides such a scheme. Consider a family of id-codes $\mathcal{F} = \{IC(v,t)\}$ with some well defined parameters (v,t).

Definition 4. *Given a batch $\mathcal{B}^u$ of arbitrary length u. Hierarchical identification based on the family of identification codes $\mathcal{F}$ is a procedure defined recursively:*

- *stopping case – if the size of the batch u is smaller or equal to some parameter v so we can use the identification code $IC(v,t) \in \mathcal{F}$, then we apply it (flat identification), otherwise*
- *recursive step – if the size of the batch u is bigger than the highest parameter v_{max} in the family $\mathcal{F}$, then it is divided into ℓ sub-batches such that $\ell \leq v_{max}$ and there is some $IC(v,t) \in \mathcal{F}$ which can be used to identify contaminated sub-batches where $\ell \leq v$ and $(t' \leq t)$.*

The hierarchical identification is denoted by $HI(\mathcal{F})$.

Hierarchical identification can be based on different collections of id-codes.There are two extreme cases:

– $\mathcal{F}$ consists of infinite sequence of id-codes,
– the family $\mathcal{F}$ is reduced to a single id-code.

No matter what is the underlying family $\mathcal{F}$, one would ask the following questions:

– What is the minimum (maximum, average) number of tests which is necessary to identify all bad signatures ?
– Given a family $\mathcal{F}$ and the number t of bad signatures in the batch, is there any procedure which minimises the number of tests ?

7.1 Hierarchical Identification with Infinite $\mathcal{F}$

Consider id-codes defined in Section 6. Each id-code can be uniquely indexed by a prime power $p > 2$. For this index, the code is $IC(p^2, p-1)$. The family

$$\mathcal{F} = \{IC(p^2, p-1) | p \text{ is the prime power}; p \neq 2\}$$

Note that $IC(p^2, p-1)$ can be used to identify up to $p-1$ bad signatures. If the number of bad signatures is $t \leq p-1$, the code will use

$$(t+1)p + 1$$

tests. If $t > p-1$, then the code fails.

Let $\#\mathcal{T}(\mathcal{F})$ be the number of tests necessary to identify all t bad signatures in a batch $\mathcal{B}^u$. Now we are trying to evaluate lower and upper bound for the number $\#\mathcal{T}(\mathcal{F})$. Assume that the size of the batch $u = p^2$ where p is a prime power. Now we choose somehow $p_1 < p$ and divide the batch $\mathcal{B}^u$ into p_1^2 sub-batches. Each sub-batch contains $\frac{u}{p_1^2}$ elements. Note that we have to consider only codes for which $t > p_1 - 1$ as otherwise the code may fail.

Let the t bad signatures be clustered into r sub-batches each containing t_i bad signatures so

$$t = \sum_{i=1}^{r} t_i$$

where $r \leq p_1 - 1$ and naturally, $t_i \leq \frac{u}{p_1^2}$. The number $\#\mathcal{T}(\mathcal{F})$ has two components:

1. the number of tests necessary to identify all contaminated sub-batches – this takes
$$\alpha = (r+1)p_1 + 1,$$
2. the number of tests necessary to identify bad signatures within the sub-batches. For a given sub-batch, we count the number of necessary tests. First we choose a prime power p_2 such that $p_2^2 \geq \frac{u}{p_1^2}$. As the sub-batch contains t_i bad signatures we need
$$\beta_i = (t_i + 1)p_2 + 1$$
tests.

The number of tests

$$\#\mathcal{T}(\mathcal{F}) = \alpha + \sum_{i=1}^{r} \beta_i$$

which after simple transformations gives

$$\#\mathcal{T}(\mathcal{F}) = (r+1)p_1 + p_2(t+r) + (r+1)$$

The number $\#\mathcal{T}(\mathcal{F})$ depends on the random parameter r and grows linearly with r so $\#\mathcal{T}(\mathcal{F})$ is smallest for $r = 1$ when all bad signatures occur in a single sub-batch. $\#\mathcal{T}(\mathcal{F})$ takes on the maximum for $r = t = p_1 - 1$. So we have the following corollary.

Corollary 6. *Given a batch $\mathcal{B}^u$ with t bad signatures. Hierarchical identification with infinite $\mathcal{F}$ will consume $\#\mathcal{T}(\mathcal{F})$ tests where*

$$2p_1 + (t+1)p_2 + 2 \leq \#\mathcal{T}(\mathcal{F}) \leq p_1^2 + 2p_1p_2 + p_1 - 2p_2.$$

7.2 Hierarchical Batching with a Single $IC(v, t)$

In some applications, one would like to keep the identification procedure as simple as possible which is using a single identification code or in other words the family $\mathcal{F}$ contains a single element. Again, knowing the number t of bad signatures in a batch $\mathcal{B}^u$, one would like to see how the number of necessary tests to identify all signatures varies (lower and upper bounds) as a function of the u and t.

Assume that $v = p^2$ and we apply the id-code $IC(p^2, p-1)$. Given a batch $\mathcal{B}^u$. There are two ways bad signatures can be identified:

- Serial identification – a batch is divided into $\frac{u}{p^2}$ sub-batches. For each sub-batch, the id-code is used. This is a serial application of flat identification.
- Hierarchical identification – a batch is divided into v sub-batches and the id-code is applied for the sub-batches and identifies the contaminated sub-batches. The process is repeated for contaminated sub-batches as many times as necessary to identify bad signatures.

Consider serial identification. Note that if a batch $\mathcal{B}^{p^2}$ is clean ($t = 0$), it takes one test to verify it. If the batch is contaminated by $t < p$ bad signatures, the identification will take $(t+1)p + t + 1$ tests. Assume that a batch $\mathcal{B}^u$ has been divided into $R = \frac{u}{p^2}$ sub-batches (if u is a multiple of p^2) among which r sub-batches are dirty and the other $R - r$ are clean. All clean sub-batches consume one test each. A dirty sub-batch $\mathcal{B}_i$ takes $(t_i + 1)(p + 1)$ tests where $\sum_{i=1}^{r} t_i = t$. So the number of tests required to identify bad signatures is

$$\frac{u}{p^2} - r + (p+1)(t+r)$$

Note that the number of tests is a random variable which ranges from $r = 1$ when all bad signatures happen to be in one sub-batch, to $r = t$ when there are t sub-batches each containing a single bad signature.

Consider the second case of hierarchical identification. To simplify our deliberations, assume that $u = p^{2j}$ for some integer j. Denote $\#\mathcal{T}(j,t)$ the number of tests needed to identify t bad signatures in a batch $\mathcal{B}^{p^{2j}}$ when the id-code is applied to the sub-batches each containing $p^{2(j-1)}$ signatures. The following recursive equation is easy to derive

$$\#\mathcal{T}(j,t) = (r+1)p + r + \sum_{i=1}^{r} \#\mathcal{T}(j-1,t_i),$$

where r is a random variable which indicates the number of contaminated sub-batches and t_i are numbers of bad signatures in the corresponding contaminated sub-batches; $i = 1, \ldots, r$.

8 Conclusions

The generic class of id-codes has been defined using the test-checking matrix A. The $(u \times v)$ matrix A determines the necessary tests. The syndrome is the binary vector which gives the test results for sub-batches defined by rows of A. The syndrome is also equal to bit-by-bit inclusive-OR of indices which correspond to bad signatures. We have investigated interaction of indices and found out that to maximise the identification capability of an id-code, one would need to choose indices of their Hamming weight equal to $v/2$.

The main construction of id-codes uses the so-called OR-checking matrix. The id-code takes a sequence of n^2 signatures and allows to identify up to $n-1$ bad signatures. The nice characteristic of the code is that the number of tests can be reduced if the batch contains less than $n-1$ bad signatures. To identify a single bad signature, it takes $2n$ tests. Any additional bad signature, adds n additional tests necessary for correct identification. There are many combinatorial structures which can be used to design id-codes. We have shown how mutually orthogonal Latin squares can be applied to construct id-codes.

We have not discussed the identification procedure of bad signatures in our id-code. The problem is far less complicated than for example in error correcting codes, mainly because the monotonicity of the Hamming weight of the syndrome. In other words, indices of bad signatures must be included in the syndrome. The implementation of this process can be done by

- checking all signatures one by one and marking those whose index collides with the syndrome,
- removing all signatures belonging to those sub-batches which have passed the test (they identified by zeros in the syndrome). In other words, all bad signatures are in the set

$$\mathcal{B} \setminus \bigcup_{\mathcal{T}(\mathcal{B}_i)=0} \mathcal{B}_i$$

where $\mathcal{B}_i$ is the sub-batch determined by the i-th row of the id-code.

Id-codes can be used directly to a contaminated batch. We called this flat identification. Alternatively, a contaminated batch can be first grouped into sub-batches and the id-code is applied to sub-batches and identifies contaminated sub-batches. This process can be done many times until bad signatures are identified. This is the hierarchical identification.

There are still many open problems. The obvious one is whether the construction given in this work is "optimal", i.e. identification of bad signatures consumes the smallest possible number of tests. Hierarchical identification allows to avoid natural limitations imposed by the size of batch and apply the id-code in hand to a batch of arbitrary length. Is there any strategy for grouping signatures into sub-batches so the number of necessary tests is minimised ?

References

1. M. Bellare, J. Garay, and T. Rabin. Fast batch verification for modular exponentiation and digital signatures. In K. Nyberg, editor, *Advances in Cryptology - EUROCRYPT'98*, pages 236–250. Springer, 1998. Lecture Notes in Computer Science No. 1403.
2. C.J. Colbourn and J.H. Dinitz, editors. *The CRC Handbook of Combinatorial Designs*. CRC Press, Boca Raton, FL, 1996.
3. J-S. Coron and D. Naccache. On the security of RSA screening. In H. Imai and Y. Zheng, editors, *Public Key Cryptography – Second International Workshop on Practice and Theory in Public Key Cryptography, PKC'99*, pages 197–203. Springer, 1999. Lecture Notes in Computer Science No. 1560.
4. J. Pastuszak, D. Michałek, J. Pieprzyk, and J. Seberry. Identification of bad signatures in batches. In H. Imai and Y. Zheng, editors, *Public Key Cryptography – Third International Workshop on Practice and Theory in Public Key Cryptography, PKC'2000*, pages 28–45. Springer, 2000. Lecture Notes in Computer Science No. 1751.