Algebraic Signatures for Scalable Distributed Data Structures

W. LITWIN
Université de Paris IX, Dauphine, France

T. SCHWARZ
Santa Clara University, USA

Abstract

Signatures detect changes to the data objects. Numerous schemes a
known, e.g., the popular hash based SHA-1 standard. We propose a no
scheme we call algebraic signatures. We use the algebraic calculus in a G
lois Field. One major consequence, new for any known signature schem
is sure detection of limited changes of parameterized size. More precise
we detect for sure any change that does not exceeds n-symbols for an
symbol signature. For larger changes, the collision probability is typica
insignificant, as for the other known schemes. We apply the algebraic sig
tures to the Scalable Distributed Data Structures (SDDSs). We filter at t
SDDS client node the updates that do not actually change the records. W
also manage the concurrent updates to data stored in the SDDS RAM buc
ets at the server nodes. We further use the scheme for the fast disk backup
these buckets. We sign our objects with 4-byte signatures, instead of 20-by
standard SHA-1 signatures that would be impractical for us. Our algebr
calculus is then also about twice as fast. We present the theory of the schem
discuss the implementation in our SDDS-2000 prototype, overview the p
formance, and directions for further work.

Keywords

Algebraic Signatures, Scalable Distributed Data Structure, File backup, Concu

1 Introduction

A signature is a string of a few bytes intended to uniquely identify t
tents of a data object (a record, a page, a file, etc.). The concept is that d
signatures prove the inequality of the contents, while identical signatu
cate equality, with high probability at least. Signatures appear therefore
potentially useful tool to detect the updates or discrepancies among th
cas [1, 2, 3, 6, 17, 22]. Their practical use required further properties, b

since the updates often follow common patterns. In a text document the c (switch) of *n* symbols usually dominates. A database record update often only relatively few bytes. Common updates should change the signatu should not lead to collisions. The collision probability should be also un low for every possible update, although no schemes can guarantee the s change for any update. Many signatures schemes with further "good pro for specific applications have been be proposed, e.g., for documents, again mission errors, malicious alterations ... [5, 8, 9]. The prominent public s is SHA-1, which provides a 160-bit secure hash signature, [20, 10]. The of a signature appeared to us useful for the management of a Scalable Dis Data Structure (SDDS). Most known SDDS schemes are a hash LH* or file, or a range partitioned RP* file [12, 13, 14], managed by the SDD prototype system available for download [4]. SDDSs are intended for or distributed databases on multicomputers, grid or P2P systems [15, 2 SDDS-2000 files reside in distributed RAM buckets for access performa rently reaching more than 100 times improvement over the disk data.

The use of signatures appeared motivated in the SDDS context as First, some applications of SDDS-2000 may need the disk back up of the from time to time. One needs then to find the only areas that changed in th since the last backup. For reasons we detail later, essentially because SDD was not initially designed for this need, the traditional dirty bit approach a impractical in our case. The signatures appeared in contrast potentially workable approach.

Next, it appears useful to sign the record updates. It was indeed obser when an application requests a record update, often it does not mean change to the record effectively occurred. Equality of the signatures betw before and after images means then that the record transfers between th cation node and the SDDS data storage server, would be useless. Likew subsequent write at the server would be the waste of time as well. Finally, appears that the signatures may also prevent the concurrent record update

However, it appeared that the properties of known signature schem the 20B size and calculus time of SHA-1, do not fit best our purpose. W duce therefore a new method that we call the algebraic signatures. Our n signature is the concatenation of n power series of Galois Field (GF) sym GF(2^8) or GF(2^{16}). While our algebraic signatures are not cryptographi cure, they exhibit a number of attractive properties. First, we may produ signatures, sufficient for our goals, e.g., 4B long. Next, the scheme is known, to the best of our knowledge, to guarantee that the objects differ symbols have guaranteed different n-symbol signatures. Furthermore, th bility that a switch or any update leads to a collision is also sufficiently may also calculate the signature of an updated object from the signatur update and from the previous signature. Finally, we may gather signatu tree of signatures, speeding up the localization of changes.

Below, we first describe more in depth our motivating SDDS needs. N
recall basic properties of a GF. Afterwards, we present our approach, we d
the implementation, and we discuss the experimental results. We conclu
directions for the further work.

2 Signatures for an SDDS

We recall that a Scalable Distributed Data Structure (SDDS) uses the serv
to store a file consisting of records or more generally objects. Records or
have a unique key and are stored on each server in buckets. The data s
implements the key-based operations of inserts, deletes, and updates as
scan queries. The application requests these operations from the SDDS
its node. The client manages the query delivery through the network to
propriate server(s) and receives the reply if any. The file scales with th
through the splits. Each split sends about half of a bucket to a new one,
ically appended. The buckets reside for the processing entirely in the dis
RAM. We apply the signatures to the disk backup of SDDS buckets an
record updates. While these were our motivating needs, others appeared
and we signal them in the Conclusion.

2.1 File Backup

We wish to backup an SDDS bucket *B* on disk. We only want to mov
parts of the bucket that are changed from the current disk copy. The tra
approach is to divide the buckets into pages of reasonably small granula
maintain a dirty bit for each page. We reset the dirty bit when the page
the disk and set it when the page is read back. We move only dirty pages
The implementation of this approach for our running prototype SDDS-2
would demand refitting a large part of the existing code that was not d
accordingly. As often, this appeared an impossible task in practice. The
code is a large software that updates the bucket in many places. Differen
functional parts of the code were produced over years by different stude
left the team since.

Another approach is to calculate a signature for each page when data
move to the disk. This computation is independent of the history of the
page and does not interfere with the existing maintenance of a data structu
is the crucial advantage in our context.

More in detail, we provide the disk copy of the bucket with a signatu
which is simply the collection of all its page signatures. Before we mov
to disk, we recalculate its signature. If the signature is identical to the ent
signature map, we do not write the page.

The slicing of the buckets into pages is somewhat arbitrary. The signat should fit entirely into RAM (or even the L2 cache using for example the macro). Smaller pages minimize transfer sizes, but increase the map and signature calculus overhead. One may expect the practical page size son between 512B and 64KB. The best choice is application depend. In any c signature calculus speed is THE challenge as it has to be small with respe disk write time. Another challenge is that the practical absence of the co to avoid an update loss. The ideal case is the zero probability of a collis is impossible or, in our case, too expensive in practice, it should be small to live with. After all, recall that when we write database updates to di only very likely, but never sure that they are actually posted. The databa generally do not bother anyway.

Presently, we implement the signature map simply as a table, since it RAM. Otherwise, the algebraic signatures allows to structure the map into ture tree. In a tree, one may compute the signature at the node from the sig of all descendents of the node. This speeds up the identification of the po the map where the signatures have changed (similar to [17] et al.) More Section 4.1.

2.2 Record Updates

We recall that an update operation only manipulates the non-key part of R. We distinguish between the *before-image* R_b, that is the contents of the client update, and the *after-image* R_a, the contents after the update and S_a denote the signatures of the before and after-image, resp. The u *normal* if R_a depends on R_b, e.g., Salary := Salary + 0.01*Sales. Th is *blind* if R_a is set independently of R_b, e.g., if one requests Salary = a house surveillance camera updates the stored image. The application r for a normal update and perhaps not for a blind one. In both cases, it is c aware whether the actual result is effectively $R_a \neq R_b$. As in the above e for unlucky salesmen in the dot-bust era, or as long as there is no burgl house.

The application nevertheless typically requests the update from the da agement system that also typically executes it. This "trusting" policy, i.e. is an update request, then it had to be the data change, characterizes in p all the DBMSs we are aware of. It is kind of surprising after all, since icy can often cost a lot. Tough times can leave thousands of salesmen sale, leading to useless transfers between clients and servers and to the processing on both nodes of thousands of records with the tuples. Lik security camera image is often a clip or movie of several Mbytes, leadi equally futile effort.

Furthermore, on the server side, several clients may attempt to read o concurrently the same SDDS record R. It is best to let every client read an without any wait. The subsequent updates should not however override ea

Our approach to this classical constraint is freely inspired by the optimisti of the concurrency control of MS-Access which is not the traditional or in database books such as [11].

In this context, the usefulness of signatures for SDDS updates comes following scheme. The application that needs R_b for a normal update re key search of R from the client. When done with its update calculus, the tion returns to the client R_b and R_a. The client computes S_a and S_b. If then the update actually did not change the record. Such updates termina client. Only if $S_a \neq S_b$ does the client send R_a and S_b to the server. Th accesses R and computes its signature S. If $S_b = S$, then the server upda $R := R_a$. Otherwise, it abandons the update. A concurrent update had to to R in the meantime, since the client read R_b and the server received it R_a. If the new update proceeded, it would override that one, making th non-serializable. The server notifies the client about the rollback, which alerts the application. The application may read R again and redo the upd

For a blind update, the application provides only R_a to the client. T computes S_a and sends the key of R_a to the server requesting S. The serv putes S and sends it to the client as S_b. From this point, the client and the se proceed as for the normal update. Calculating and sending S alone as S_b avoids the transfer of R_b to the client. It may avoid further the useless of R_a to the server. These can be substantial savings, e.g., for the surv images.

The scheme does not need locks. Also, as we have seen, the signature saves the useless record transfers. Besides, neither the key search, nor th or deletion need the signature calculus. Hence, none of these operation the concurrency management overhead. All together, the degree of conc can be potentially high. The scheme roughly corresponds to the R-Co isolation level of the SQL3 standard. Its properties make it attractive applications that do not need transaction management. Especially, if sear the predominant operation, as one considers in general for an optimistic s

The scheme does not store signatures. Hence, the storage overhead ca terestingly strictly zero. This is not possible for timestamps, probably use Access, although that overhead is usually negligible, hence perfectly ac in practice. In fact, it can still be advantageous to vary the signature sch storing the signatures with their records. As we show later, the storage co server can be then also usually negligible, of only about 4B per signatu client sends in this case also S_a to the server which stores it in the file if it accepts the update. When the client requests R it gets it with S. If th requests S alone, the server simply extracts S from R, instead of dynamic culating it. All together, one saves the S_b calculus at the client and tha the server. Also, and more significantly perhaps in practice, the signature becomes entirely deported at the client. Hence, it is entirely parallel am concurrent clients. This can enhance the update throughput even further.

Whether one stores the signature or not, the speed of the signature ca clearly *the* challenge again. Since a record key search in or insert into a reaches the speed of 0.1 ms at present, the record signature calculus time be longer than dozens of microseconds in practice. Another challenge again, the total or at least practical absence of the collisions, to avoid any loss.

3 Galois Fields

A Galois field (GF) is finite field. Addition and multiplication in a GF a ciative, commutative, and distributive. There are neutral elements called z one for addition and multiplication respectively, and there exist inverse e regarding addition and multiplication. We denote by GF(2^f) a GF over t all binary strings of a certain length f. GF(2^8) and GF(2^{16}) are our main Their elements are respectively byte and 2-byte strings.

We identify each binary string with a binary polynomial in one for known x. For example, we identify the string 101001 with the polynom x^3+1. We further associate a *generator polynomial* $g(x)$ with the GF. T polynomial of degree f that cannot be written as a product of two other mials other than the trivial result of a multiplication of 1 with itself.

The addition of two elements in our GF is that of their binary poly In practice, the sum of two strings is the XOR of the strings. The pr two elements is the binary polynomial obtained by multiplying the two polynomials and taking the remainder modulo $g(x)$. There are several implement this calculus. We use the logarithmic multiplication method we soon. It uses the *primitive* elements of a GF with s elements, which are e with the following properties. The *order* of a non-zero element α, ord(α smallest exponent non-zero i such that $\alpha^i = 1$. All non-zero elements in a a finite order. An element α is primitive, if $\text{ord}(\alpha) = s-1$. It is well known any given primitive element α, all the non-zero elements in the field are powers α^i,, each with a uniquely determined exponent $i \in \{0, \ldots s-1\}$. usually has several primitive elements. In particular, any α^i is also a p element if i and $s-1$ are coprime, i.e., without non-trivial factors in c Our GFs contain 2^f elements, hence the prime decomposition of 2^f-1 contain the prime 2. For our basic values of $f = 8, 16$, 2^f-1 has only few hence there are relatively many primitive elements. For example, for f count 127 primitive elements or roughly half the elements in the GF.

We fix one primitive element α. Every non-zero element β is a power $\beta = \alpha^i$, we call i the logarithm of β with respect to α and write $i = \log_{\alpha}$ we call *beta* the antilogarithm of i with respect to α and write $\beta =$ ant The logarithms are uniquely determined if we choose i to be $0 \le i \le 2^f$ set $\log(0) = -\infty$.

The multiplication is now given by the following formula which uses modulo $2^f - 1$:

$$\beta \cdot \gamma = \text{antilog}_{\alpha}(\log_{\alpha}(\beta) + \log_{\alpha}(\gamma)).$$

We implement GF multiplication on this basis as follows. We create one logarithms of size 2^f symbols. We also create another one for antilogar size $2^f \cdot 2$. That table has two copies of the basic antilog table. It accom indices up to size $2f \cdot 2$ avoiding the slower modulo calculus of the form $f = 8, 16$ both tables may fit also into the L1 or L2 cache of current pr (not all for $f = 16$). We also check for the special case of one of the o being equal to 0. All together, we obtain the following simple C-pseudo-

```
GFElement mult(GFElement left, GFElement right) 
   if(left==0||right == 0) return 0;
   return antilog[log[left]+log[right]];}
```

In terms of Assembly instructions, the typical execution costs of the boo sub-program are two comparisons, four additions (three for table-look-u memory fetches and the return statement.

4 Algegraic Signatures

4.1 Basic Properties

We call a page a string of l symbols $p_i; i = 0, \ldots l-1$. In our case, the sy are bytes or two-byte words. The symbols are elements of a Galois field with $f = 8$ or $f = 16$. We assume that $l < 2^f - 1$.

Let $\vec{\alpha} = (\alpha_1, \ldots \alpha_n)$ be a vector of different non-zero elements of the call $\vec{\alpha}$ the *n-symbol signature base* or simply *base*. The *n-symbol sign* P based on $\vec{\alpha}$ is the vector $\vec{\text{sig}}_{\vec{\alpha}}(P) = (sig_{\alpha_1}(P), sig_{\alpha_2}(P), \ldots, sig_{\alpha_n}(P)$ for each α we set $sig_{\alpha}(P) = \sum_{i=0}^{l-1} p_i \alpha^i$. We call each coordinate of $\vec{\text{sig}}$ *component signature.*

We have not completely investigated what choice of the coordinates is for other applications. We primarily use the base $\vec{\alpha} = (\alpha, \alpha^2, \alpha^3, \ldots \alpha^n)$ w $2^f - 1$ and primitive α and write $\text{sig}_{\alpha,n}$ instead of $\vec{\text{sig}}_{\vec{\alpha}}$ The collision pr of $\text{sig}_{\alpha,n}$ is at best 2^{-nf}. This is probably insufficient for $n = 1$.

We are also interested in the signature $\text{sig}^{(2)}_{\alpha,n} = \vec{\text{sig}}_{\vec{\alpha}}$ with $\vec{\alpha} = (\alpha, \alpha$ $\ldots \alpha^{2n-2})$, where the base coordinates are all primitive.

The basic new property of $\text{sig}_{\alpha,n}$ is that any change of up to n symbo P changes the signature *for sure.* This is our primary concern in this schem formally we stay this property as follows.

Proposition 1 *Provided the page length l is $l < \text{ord}(\alpha) = 2^f - 1$, $\text{sig}_{\alpha,n}$ c any change of up to n symbols per page.*

Proof: As α is primitive and our GF is GF(2^f) we have ord$(\alpha) = 2^f - 1$. that the file symbols at locations $i_1, i_2, \dots$ in P have been changed, but signatures of the original and the altered file are the same. Call d_ν the di between the respective symbols in position $i - \nu$. The difference of the co signatures is then:

$$\sum_{\nu=1}^{n} \alpha^{i_\nu} d_\nu = 0 \qquad \sum_{\nu=1}^{n} \alpha^{2\cdot i_\nu} d_\nu = 0 \qquad \dots \qquad \sum_{\nu=1}^{n} \alpha^{n\cdot i_\nu} d_\nu = 0.$$

The d_ν values are the solutions of a homogeneous linear system

$$\begin{pmatrix} \alpha^{i_1} & \alpha^{i_2} & \alpha^{i_3} & \alpha^{i_4} & \dots & \alpha^{i_n} \\ (\alpha^{i_1})^2 & (\alpha^{i_2})^2 & (\alpha^{i_3})^2 & (\alpha^{i_4})^2 & \dots & (\alpha^{i_n})^2 \\ (\alpha^{i_1})^3 & (\alpha^{i_2})^3 & (\alpha^{i_3})^3 & (\alpha^{i_4})^3 & \dots & (\alpha^{i_n})^3 \\ (\alpha^{i_1})^4 & (\alpha^{i_2})^4 & (\alpha^{i_3})^4 & (\alpha^{i_4})^4 & \dots & (\alpha^{i_n})^4 \\ \dots & \dots & \dots & \dots & \dots & \dots \\ (\alpha^{i_1})^n & (\alpha^{i_2})^n & (\alpha^{i_3})^n & (\alpha^{i_4})^n & \dots & (\alpha^{i_n})^n \end{pmatrix} \cdot \begin{pmatrix} d_1 \\ d_2 \\ d_3 \\ d_4 \\ \vdots \\ d_n \end{pmatrix} =$$

The coefficients in the first row are all different, since the exponents $i_\nu <$ The matrix is of Vandermonde type, hence invertible. The vector of dif $(d_1, d_2 \dots d_n)^t$ is thus the zero vector. This contradicts our assumption. Th detects any up to n-symbol change. CQFD

Notice that Prop. 1 trivially holds for $\text{sig}^{(2)}_{\alpha,n}$ with $n \leq 2$. $\text{sig}_{\alpha,n}$ has possible behavior of for changes limited to n symbols. An application can possibly change up to $l > n$ symbols. We now prove that $\text{sig}_{\alpha,n}$ still exh low collision probability typically expected from a signature schema.

Proposition 2 *Assuming a page length $l <$ ord(α) and every possible p tent equally likely, the signatures $\text{sig}_{\alpha,n}(P_1)$ and $\text{sig}_{\alpha,n}(P_2)$ of two differe P_1 and P_2 collide (coincide) with a probability of 2^{-nf}.*

Proof: The n-symbol signature is a linear mapping between the vecto GF$(2^f)^l$ and GF$(2^f)^n$. This mapping is an epimorphism, i.e., every ele GF$(2^f)^n$ is the signature of some page, an element of GF$(2^f)^l$. Consider ϕ, which maps every page with all but the first n elements equal to ze signature. Thus, $\phi : \text{GF}(2^f)n \rightarrow \text{GF}(2f)l, (x1,,xn) \rightarrow \text{sig}_{\alpha,n}((x_1,\dots,xn,($ and:

$$\phi((x_1, x_2, \dots x_n)) = \begin{pmatrix} \alpha & \alpha^2 & \alpha^3 & \alpha^4 & \dots & \alpha^n \\ \alpha^2 & (\alpha^2)^2 & (\alpha^3)^2 & (\alpha^4)^2 & \dots & (\alpha^n)^2 \\ \alpha^3 & (\alpha^2)^3 & (\alpha^3)^3 & (\alpha^4)^3 & \dots & (\alpha^n)^3 \\ \alpha^4 & (\alpha^2)^4 & (\alpha^3)^4 & (\alpha^4)^4 & \dots & (\alpha^n)^4 \\ \dots & \dots & \dots & \dots & \dots & \dots \\ \alpha^n & (\alpha^2)^n & (\alpha^3)^n & (\alpha^4)^n & \dots & (\alpha^n)^n \end{pmatrix} \cdot \Bigg($$

The matrix is again of Vandermonde type, and hence invertible. This imp every possible vector in $\mathrm{GF}(2^f)^n$ is the signature of a page with all but n symbols equal to zero, and of only one such page. Consider now an a vector $\vec{s}$ in $\mathrm{GF}(2^f)^n$. Each page of form $(0,\ldots 0, x_{n+1}, x_{n+2}, \ldots x_l)$ has so tor $\vec{t}$ in $\mathrm{GF}(2^f)^n$ as its signature. For any $\vec{s}$ and $\vec{t}$ there is then exactly o $(x_1,\ldots x_n, 0, 0, \ldots 0)$ has therefore signature $\vec{s}$. Thus, the number of pages t signature $\vec{s}$ is that of all pages of form $(0,\ldots 0, x_{n+1}, x_{n+2}, \ldots x_l)$. There ar such pages. There are furthermore 2^{fl} pages in total. A random choic pages leads thus to the same signature $\vec{s}$ with probability $2^{f(l-n)}/2^{fl} = 2$ suming that all pages are equally likely to be selected, our proposition CQFD.

Notice that Proposition 2 also characterizes $\mathrm{sig}^{(2)}_{\alpha,n}$ for $n \leq 2$. Next, ou tures are called *algebraic* and claim at least by name some algebraic pr Here is one motivating property of $\mathrm{sig}_{\alpha,n}$. Its gains practical importance f subject to very localized and small changes. This case is typical in da where attributes have typically rather few symbols. We show that one n update the $\mathrm{sig}_{\alpha,n}$ signature from the changed symbols and the before si This can clearly speed up the calculation of signatures over a complete re tion as is necessary for SHA1. Formally:

Proposition 3 *Let us change page* $P = (p_0, p_1, \ldots p_{l-1}$ *to page* P' *wher place the symbols starting in position* r *and ending with position* $s-1$ *string* $q_r, q_{r+1}, \ldots q_{s-1}$. *We call the string* $\Delta = (\delta_0, \delta_1, \ldots, \delta_{s-r-1})$ *with* δ_i q_{r+1} *the* Δ *string. Then for each* α *in our base* $\vec{\alpha}$ *we have*

$$sig_\alpha(P') = sig_\alpha(P) + \alpha^r sig_\alpha(\Delta)$$

Proof: The difference between the signatures is $\mathrm{sig}_\alpha(P') - \mathrm{sig}_\alpha(P) = \Sigma$ $p_i)\alpha^i = \alpha^r \left(\sum_{i=r}^{s-1}(q_i - p_i)\alpha^{i-r}\right) = \alpha^r \left(\sum_{i=r}^{s-1} \delta_{i-r}\alpha^{i-r}\right) = \alpha^r \sum_{\nu=0}^{s-r-1} \delta_\nu \alpha^i =$ CQFD.

We finish the section with a proof of the practicality of the $\mathrm{sig}^{(2)}_\alpha$ schen context of the popular switch (cut / paste) operation. Prop. 1 shows sure d for any switch of length $\leq n/2$. In many applications such as text editing, ing larger pieces of text are common. Prop. 2 does not cover this case. seek to determine and minimize the collision probability in that context The base $\vec{\alpha} = (\alpha, \alpha^2, \alpha^4 \ldots \alpha^{2n})$ has coordinates of largest possible orde unlike the base $(\alpha, \alpha^2, \alpha^3, \ldots \alpha^n)$. Intuitively therefore, $\mathrm{sig}^{(2)}_\alpha$ appears pr to sig_α and the following proposition confirms this in the case of cut a operations.

Proposition 4 *Assume an arbitrary page* P *of length* $> 2n$ *and three ind* t *of appropriate sizes, Figure 1. Assume a base* $\vec{alpha}$ *whose coordinates order larger than the length of the page. Cut a string* T *of length* t *beginn*

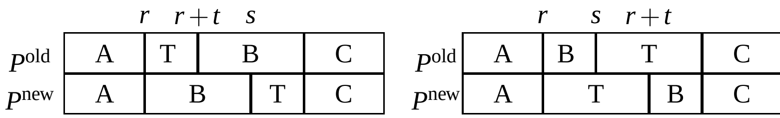

Figure 1: Illustration of the cut and paste operation.

position r and move it to position s in P. Assuming any T to be equally li probability that $\mathrm{sig}_{\vec{\alpha}}(P)$ *changes is* 2^{-nf}.

Proof: Either T or the rest of the page contain at least n symbols, we the latter, the former being analogous. Without loss of generality, we a forward move of T within the file from position r to position s. A backwa just undoes this operation and thus has the same effect on the signature defines the name for the regions of the block and makes a spurious case dis depending on whether $r+t<s$ or not. For any coordinate α in the b signature of the "before" page (the top scheme for both situations) is

$$\mathrm{sig}_\alpha(P^{\mathrm{old}}) = \mathrm{sig}_\alpha + \alpha^r \mathrm{sig}_\alpha(T) + \alpha^{r+t}\mathrm{sig}_\alpha(B) + \alpha^{s+t}\mathrm{sig}_\alpha(C).$$

The after page signature is

$$\mathrm{sig}_\alpha(P^{\mathrm{new}}) = \mathrm{sig}_\alpha + \alpha^r \mathrm{sig}_\alpha(B) + \alpha^{s}\mathrm{sig}_\alpha(T) + \alpha^{s+t}\mathrm{sig}_\alpha(C).$$

The difference of the two signatures is

$$\begin{aligned}
\mathrm{sig}_\alpha(P^{\mathrm{new}}) - \mathrm{sig}_\alpha(P^{\mathrm{old}}) &= \alpha^r\mathrm{sig}_\alpha(T) + \alpha^{r+t}\mathrm{sig}_\alpha(B) + \alpha^r\mathrm{sig}_\alpha(B) + \alpha \\
&= (\alpha^r + \alpha^s)\mathrm{sig}_\alpha(T) + (\alpha^r + \alpha^{r+t})\mathrm{sig}_\alpha(B) \\
&= \alpha^r\left((1+\alpha^s)\mathrm{sig}_\alpha(T) + (1+\alpha^t)\mathrm{sig}_\alpha(B)\right).
\end{aligned}$$

This expression is zero only if the right hand side, or the following exp where we use γ_i as an abbreviation, is zero:

$$(1+\alpha_i^s)(1+\alpha_i^t)^{-1}\mathrm{sig}_{\alpha_i}(T) + \mathrm{sig}_{\alpha_i}(B) =$$

$$(1+\alpha_i^s)(1+\alpha_i^t)^{-1}\mathrm{sig}_{\alpha_i}(T) + \sum_{\nu=n}^{|B|-1} b_\nu\alpha_i^\nu \sum_{\nu=0}^{n-1} b_\nu\alpha_i^\nu = \gamma_i + \sum_{\nu=0}^{n-1} b_\nu\alpha_i^\nu$$

We now fix the whole situation with the exception of the first n symbols i change in signatures is

$$\left(\gamma_0 + \sum_{\nu=0}^{n-1}\alpha_0^\nu b_\nu, \gamma_1 + \sum_{\nu=0}^{n-1}\alpha_1^\nu b_\nu, \ldots\right) = (\gamma_0, \gamma_1, \ldots) + \left(\sum_{\nu=0}^{n-1}\alpha_0^\nu b_\nu, \sum_{\nu=0}^{n-1}\alpha_1^\nu b\right.$$

which is zero if and only if

$$(\gamma_0, \gamma_1, \ldots) = \left(\sum_{\nu=0}^{n-1} \alpha_0^{\nu} b_{\nu}, \sum_{\nu=0}^{n-1} \alpha_1^{\nu} b_{\nu}, \ldots \right)$$

The left hand side is a linear mapping in the $(b_0, b_1, \ldots)$, which has a matr
invertible because it has a Vandermonde type determinant. Therefore, the
only one combination of $(b_0, b_1, \ldots b_{n-1})$ that is mapped by the mapping
right hand vector. This combination will be attained by a randomly picke
probability 2^{-nf}. CQFD

At this stage of our research, the choice of $\text{sig}_{\alpha,n}^{(2)}$ appears only as a
between smaller probability of collision for possibly frequent updates (
here), and the zero probability of collision for updates up to any n symb
are able only to conjecture that there is a α in GF(2^8) or GF(2^{16}) fo
Propositions 1 and 2 holds for $\text{sig}_{\alpha,n}^{(2)}$ with $n > 2$. We did not pursue th
tigation further. For our needs, $n = 2$ for GF(2^{16}) was sufficient (Section
$\text{sig}_{\alpha,2}^{(2)} = \text{sig}_{\alpha,2}$ the properties of both schemes coincide anyway.

4.2 Compound Algebraic Signatures

Our signature schemes keep the property of sure detection of n-symbol c
long as the page size in symbols is at most $2^f - 2$. For $f = 16$, the lim
page size is almost 128 KB. Such granularity suffice for our purpose. C
have many pages in an SDDS bucket that can reach, e.g., 256 MB for SDD
The collections of the signatures in the bucket may be seen as a vector.
it compound signature (of the bucket). More generally, we qualify a co
signature of m pages, as m-fold. The the signature map of Section 2.1 imp
a compound signature.

The practical interest of the compound signatures stretches beyond o
vating cases. We may usefully apply the concept as an alternative to a sign
an area A not exceeding the limit of $\text{ord}(\alpha) - 1$. To use an m-fold signatu
for instance, one may divide A into equally sized pages each provided wit
We locate then for sure and with granularity of l/m any change of up to n
with a priori unknown location (hence Proposition 3 does not apply). T
with respect to $\text{sig}_{\alpha,n}(A)$, i.e., $\text{sig}_{\alpha,n}$ over the entire A with granularity t
is mainly the about m times larger storage overhead. In practice, one ca
for m leading to a reasonable compromise. Notice that a yet alternative ch
the m times larger overhead if acceptable, is to enhance the sure change d
resolution to mn symbols anywhere in A, using $\text{sig}_{\alpha,mn}(A)$.

For larger m, we can exploit the algebraic properties of the m-fold s
scheme by implementing signature maps as trees to speed up the sear
changed $\text{sig}_{\alpha,n}$. As we show below, with our schemes, we may algeb

i.e., without reexamining the pages themselves compute the higher-leve tures (unlike for more traditional signature schemes we are aware of). If changes, we may update the higher level once again only algebraically. A capabilities of compound signatures can be of obvious interest to our SD backup application.

The following proposition proves the algebraic properties we discus area partitioned into two pages. Those can be furthermore of different siz is sometimes a useful capability as well, e.g., when A starts with a relative index of the data that follow in A. It generalizes trivially to any larger m. pages of different sizes as well.

Proposition 5 *Consider two pages P_1 and P_2 of length l and m, $l+m \leq$ concatenated into a page (area) $P_1|P_2$. Then $\mathrm{sig}_{\alpha,n}$ of the concatenated p be calculated from the component pages by the formulae*

$$\mathrm{sig}_{\alpha^i}(P_1|P_2) = \mathrm{sig}_{\alpha^i}(P_1) + \alpha^{il}\mathrm{sig}_{\alpha^i}(P_2).$$

Proof Assume that $P_1 = s_1, s_2, \ldots s_l$ and that $P_2 = s_{l+1}, s_{l+2}, \ldots s_{l+m}$. The

$$\mathrm{sig}_\beta(P_1|P_2) = \sum_{\nu=1}^{l+m} s_\nu \beta^\nu = \sum_{\nu=1}^{l} s_\nu \beta^\nu + \sum_{\nu=l+1}^{l+m} s_\nu \beta^\nu = \sum_{\nu=1}^{l} s_\nu \beta^\nu + \beta^l \sum_{\nu=1}^{m} s_{\nu+l}$$
$$= \mathrm{sig}_\beta(P_1) + \beta^l \cdot \mathrm{sig}_\beta(P_2$$

for any β in the GF. CQFD Proposition 5 applies to both $\mathrm{sig}_{\alpha,n}$ and sig gether, all propositions we have formulated prove the potential of our s schemes. They have further algebraic properties we are currently investig

5 Experimental Implementation

5.1 Calculus Tuning

We can tune the signature calculus. First, we can interpret the page sym rectly as logarithms. This saves a table look-up. The logarithms range to $2^f - 2$ (inclusively) with the additional value for log(0). One can set to $2^f - 1$. Next, the signature calculations form the product with α^i. has i as the logarithm. One does not need to look this value up neither. lowing pseudo-code for $\mathrm{sig}_{\alpha,1}$ applies these properties. It uses as param address of an array representing the bucket and the size of the bucket. T stant `TWO_TO_THE_F` is 2^f. The type GFElement is an alias for the app integer type.

```
GFElement signature(GFElement *page, int pageLeng
   GFElement returnValue = 0;
```

```
   for(int i=0; i< pageLength; i++)
      if(page[i]!=TWO_TO_THE_F-1)
           returnValue ^= antilog[i+page[i]];
   return returnValue;
}
```

The application to $\mathrm{sig}_{\alpha,n}$ is easy. In our file backup application, the bucke contains several pages so we typically calculate the compound signature. this calculus, we should consider the best use of the processor caches, i.e. L2 caches on our Pentium machines. It seems advantageous to exploit th lines on the log table. Then, it may be gainful to first loop upon the cal sig_{α} for all the pages, then move to sig_{α^2} and so on. Our experiments co this intuition.

5.2 Experimental Performance

We have implemented the motivating applications with the $\mathrm{sig}_{\alpha,1}$ schem experimental analysis. The testbed configuration consisted from 1.8 G tium P4 nodes and from 700 Mhz Pentium P3 nodes over a 100 Mbs E One implementation concerned the signature calculus schemes alone wi lated data. The experiments examined variants of the $\mathrm{sig}_{\alpha,n}$ calculus with to implementation issues and some differences with respect to the basic We have also experimented the $\mathrm{sig}^{(2)}_{\alpha,n}$ whose calculation time turned out same. Finally, we have ported the fastest algorithm of $\mathrm{sig}_{\alpha,n}$ calculus to 2000. In both cases, we have divided the bucket into the pages of size 16 a 4-byte signature per page. This choice appears to be a reasonable com between the signature size, hence its calculation time, and the overall probability of order 2^{-32}, i.e. over $4 \cdot 10^{-9}$. For record updates, we use t signature size, but the record size is 100 bytes. If we had used SHA-1, t head would be 20 bytes per page or record, [20]. Records had about 100 our experiments.

Internally, the bucket in SDDS-2000 has a RAM index as it is structu a RAM B-tree. The index is small, a few KB at largest. Bucket size pa not make sense there. We set up for the page size of 128 B for the inde for the record updates, we set up for the signature calculus on-the-fly on seemed the most flexible choice, but we could alternatively store the sign each record, avoiding its calculation. The actual computation took place the updates. Inserts were not affected.

The analysis of the experiments with the actual SDDS-2000 implem is presented full in [19]. The main results are as follows. The stand-alon iments showed, somehow surprisingly, a large variation of the calculat depending on the data symbols. The reason seemed to be the influenc caches L1 and L2. For a given page size, the calculus time was linear w

$sig_{\alpha,n}$ used. The actual calculus times of $sig_{\alpha,2}$ as actually put into SDI
was of 20-30 ms per 1 MB of RAM bucket, manipulated as a mapped
SHA-1, our tests showed about 50-60 ms. The $sig_{\alpha,2}$ calculation time
as wished in the order of dozens of microseconds for the index page or a
This timing was linear with the bucket or record size, and, also someh
prisingly, rather stable regardless of the algebraic signature scheme tes
calculus time was smaller for a larger page: 64 KB versus 16 KB. It is p
due to the better cache use. The actual transfer time of 1 Mbyte of RAI
disk is about 300 ms. Thus the backup using our signature scheme off
expected gains. Likewise, our signature based record update managemen
a practical solution as well.

For both page sizes, calculation in GF(2^{16}) was faster than in GF(2
despite the fact that the logarithm table of the latter could entirely enter the
cache of each of our machines, accelerating thus the calculus. The former
used in turn more effectively the 4B words. Being faster, GF(2^{16}) was
choice for SDDS-2000.

6 Conclusion

Our schemes possess properties novel to signature schemes, namely
detection of limited changes of parameterized size, and of algebraic op
over the signatures themselves. Together with the high probability of dete
any change, including switches, small overhead, and fast calculus, our a
proved itself to be useful for our motivating SDDS needs of bucket back
record updates. The experimental fine-tuning of the implementation of th
ture calculus allowed us finally to successfully add the $sig^2_{\alpha,n}$ scheme to
2000 system.

Among future research directions, one concerns the applications of t
braic properties of the schemes. One direction currently investigated is the
text string parallel search (scan) in the non-key fields of records at SSDS
While the need is classical and many algorithms are widely used for yea
Boyer-Moore or Knuth, the algebraic signatures lead to a new approach
teresting feature is that the client can send to each server only the few-b
signature and the length of the string to search, instead of the entire stri
haps long, hence costly to transmit, especially if the SDDS client should
to many servers. The server then compares only the incoming signature
of the actually examined string within the searched record. If the matc
successful, and we should move forward in the record, typically by one
the signature of the new string to examine is algebraically recalculated f
previous one. The calculus uses the properties discussed in Section 4.1. T
is much faster than if one had to recalculate it entirely. This would be the
any other signature scheme we are aware of (making any such attempt us

practice).

Beyond, one should determine further algebraic properties of the s
The conjecture of sure detection of changes also by $\mathrm{sig}^2_{\alpha,n}$ scheme rema
proved or disproved. Variants of the basic schemes should be studied. Th
signature trees for computing the compound signatures and explore the s
maps is an open research area. Finally, we did not explore the Prefetch
providing perhaps further savings to the calculus time through better L1
cache management.

Beyond these goals, one can apply our schemes to the automatic file
in presence of several files sharing an SDDS server whose RAM becam
ficient for all the files simultaneously, [16]. Next, the signatures appear
useful tool for the cache management at the SDDS client, allowing to
cache and server data synchronized. There is also an interesting relation
tween the algebraic signatures and the Reed-Salomon parity calculus we
the high-availability SDDS LH*RS scheme, [14]. GFs are the common b
it appears that the signatures may help preserving the mutual consistency
and parity records in presence of lost messages.

Our techniques should help also other database needs. Especially,
pear attractive for a RAM-DBS that typically needs the RAM data im
the disk as well. Very large Gbyte RAMs are now widely available, a
enhanced performance of such DBSs increasingly attractive with respect
of traditional disk-based DBSs, [21]. Likewise, our signature based recor
calculus at the SDDS client, should provide similar advantages for a clie
based DBS architecture in general. Interesting possibilities appear furthe
transactional concurrency control, beyond the avoidance of the lost upda

Acknowledgments

We thank Lionel Delafosse, Jim Gray and Peter Scheuermann for frui
cussions. This work was partly supported by the research grants from M
Research, and from the European Commission project ICONS project
2001-32429 and the SCU IBM grant 41102-COEN-RSCH-IG-IG09.

References

[1] K. A. S. Abdel-Ghaffar, and A. El-Abbadi. Efficient Detection of C
Pages in a Replicated File. *ACM Symp. Distributed Computing*,
1993.

[2] D. Barbara, H. Garcia-Molina, and B. Feijoo. Exploiting Symme
Low-Cost Comparison of File Copies. *Proc. Int. Conf. Distributed*
ing Systems, 471-479, 1988.

[3] D. Barbara, and R. J. Lipton. A class of Randomized Strategies f Cost Comparison of File Copies. *IEEE Trans. Parallel and Distribu tems* 2(2):160-170, 1991.

[4] Ceria Web site. http://ceria.dauphine.fr.

[5] C. Faloutsos, and S. Christodoulakis. Optimal Signature Extraction formation Loss. *ACM Trans. Database Systems* 12(5):395-428, 198

[6] W. Fuchs, K. L. Wu, and J. A. Abraham Low-Cost Comparison an nosis of Large, Remotely Located Files. *Proc. Symp. Reliability Dis Software and Database Systems* 67-73, 1986.

[7] D. Gollmann. Computer Security. Wiley, 1999.

[8] J.-K. Kim, and J.-W. Chang. A new parallel Signature File Method cient Information Retrieval. *CIKM 95* 66-73, 1995.

[9] S. Kocberber, and F. Can. Compressed Multi-Framed Signature F Index Structure for Fast Information Retrieval. *SAS 99* 221-226, 19

[10] C. Kaufman, R. Perlman, and M. Speciner. Network Security: Priva munication in a Public World. Prentice Hall, 2002.

[11] P. M. Lewis, A. Bernstein. M. Kifer. Database and Transaction Pro Addison & Wesley, 2002.

[12] W. Litwin, M-A. Neimat, and D. Schneider. RP* : A Family o Preserving Scalable Distributed Data Structures. *20th Intl. Conf Large Data Bases (VLDB)*, 1994.

[13] W. Litwin, M-A. Neimat, and D. Schneider. LH*: A Scalable Dis Data Structure. *ACM Transactions on Database Systems ACM-TOI* 1996.

[14] W. Litwin, and T Schwarz. LH*RS: A High-Availability Scala tributed Data Structure using Reed Solomon Codes. *ACM-SIGMO* 2000.

[15] W. Litwin, T. Risch, and T Schwarz. An Architecture for a Scala tributed DBS :Application to SQL Server 2000 *2nd Intl. Worksho operative Internet Computing (CIC 2002), Hong Kong* , 2002.

[16] W. Litwin, P Scheuermann, and T. Schwarz. Evicting SDDS-2000 in RAM to the Disk. CERIA Res. Rep. 2002-07-24, U. Paris 9, 200

[17] J. Metzner. Structure for Large Remotely Located Data Files. *IEE actions on Computers*, C-32(8), 1983.

[18] J. Michener, T. Acar. Managing System and Active-Content Integrit *puter*, 108-110, July 2000.

[19] R. Mokadem. Storing Data Using Signatures in SDDSs (Stoc Donnes en utilisant les Signatures dans les SDDS). *Mmoire DEA, 9, Dauphine*, 2002.

[20] Natl. Inst. of Standards and Techn. Secure Hash Standards. FI 180-1, Apr. 1995.

[21] Y. Ndiaye, A. Diène, W. Litwin, and T. Risch. AMOS-SDDS: A Distributed Data Manager for Windows Multicomputers. *14th Intl. ence on Parallel and Distributed Computing Systems – PDCS 2001*

[22] T. Schwarz, B. Bowdidge, W. Burkhard. Low Cost Comparison of F *Conf. on Distr. Comp. Syst., (ICDCS 90)*, 196-201, 1990.

Witold Litwin is at the Centre des Études et de Recherches en Informatique A Université Paris 9 Dauphine Pl. du Mal. de Lattre de Tassigny 75016 Paris FR mail: Witoldlitwin@dauphine.fr

Thomas Schwarz is with the Department of Computer Engineering at Santa C versity. Santa Clara University Santa Clara, CA 95053 USA E-mail: tjschwarz@