Homomorphic Signatures for Digital Photographs

Rob Johnson[***], Leif Walsh[***], and Michael Lamb[†]

Stony Brook University

Abstract. We describe two homomorphic signature schemes for digital photographs such that an intermediate party in possession of a signed photograph can construct a scaled, cropped, and lossily compressed version of the photograph along with a new, valid signature, *without knowing the private signing key*. In other words, our signature schemes are simultaneously homomorphic with respect to cropping, scaling, and JPEG-like compression. Unlike prior ad-hoc schemes for photographic signatures, our first scheme is provably secure and quite practical. For example, a scaling-homomorphic signature scheme using our techniques requires less than 100KB of signature data for typical digital photographs. Our second signature scheme has weaker security but reduces typical signature sizes to 15KB. Both schemes extend naturally to authenticate movies and other digital media and use novel, multi-dimensional variations of Merkle hashing and GGM trees related to constructions used in computational geometry that may be of independent interest.

1 Introduction

We present homomorphic signature schemes that are simultaneously homomorphic with respect to cropping, scaling, and JPEG-like lossy compression. With such a signature, anyone possessing a signed digital image can perform any combination of these image edits and, by performing corresponding operations on the signature, create a new scaled, cropped, compressed image with a valid signature, without knowing the private signing key. Our signature schemes are efficient, requiring only one public-key operation and, for typical uses, under 100KB of signature data for digital photographs under 16 megapixels. Unlike previous ad-hoc attempts to create homomorphic signatures for digital photographs, our primary signature scheme is provably secure. The second signature scheme we present sacrifices some security, but reduces signature sizes to 15KB for typical digital images. Our homomorphic signatures extend easily to higher-dimensional data so, for example, we could create a movie signature scheme that is also homomorphic w.r.t. scene cuts and deletions.

Digital photographs are ubiquitous, so signature schemes homomorphic with respect to common image operations could have numerous applications. A digital camera equipped to produce a signature of each photograph it creates would enable photographers to prove that their photographs are real and unaltered. These photographs could then be cropped and scaled as appropriate, and the final viewer could verify that the photograph they see is authentic. Scientific journals could require such signatures on photographic evidence in their submissions, preventing fraud such as the human cloning forgeries published in Science[29]. Online news sites could use such signatures to provide an end-to-end proof that photos accompanying their new stories are real, preventing photo-journalism fraud such as Reuters' digitally altered 2006 Lebanon war photographs[31]. Police could use these signatures to prove that crime-scene photos or security camera footage is authentic.[1]

[*] rob@cs.sunysb.edu

[**] This research was supported by National Science Foundation grant CNS 0627645.

[***] rlwalsh@ic.sunysb.edu

[†] mike@datagrok.org

[1] This technology can not prevent all kinds of photographic forgery, but it can make forgeries significantly more expensive to produce.

Our signature scheme follows the redactable signature framework of Johnson, et al[10], but uses novel multi-dimensional variants of Merkle hashing (Section 4) and GGM trees (Section 5). Standard Merkle hashing computes a value, h for a data vector, x, such that, by presenting $O(\log(|x| - |x'|))$ witnesses, one can prove that some contiguous subvector x' of x was part of the original data used to compute h. In our extension, one can prove that some hyperrectangular submatrix A' was part of an original matrix A used to compute the hash, although more witnesses are required. For example, in 2-dimensions, we require $O((\log HW)^2)$ witnesses, where W and H are the width and height of the original matrix. This hashing scheme may be of independent interest. The GGM PRNG construction generates a sequence of n pseudorandom outputs such that one can reveal any contiguous subsequence by only communicating $\log n$ PRNG seeds. We present two 2-dimensional analogs. The first uses a GGM tree and a space-filling curve to generate an $H \times W$ matrix of pseudorandom outputs such that any $h \times w$ submatrix of outputs can be revealed by transmitting $O(h+w)$ PRNG seeds. The second construction is not a PRNG, but it suffices to construct cropping-, scaling-, and compression-homomorphic signature schemes that are secure against an adversary that makes 1 signing oracle query and it reduces the number seeds needed to $O(\log hw)$. We then use these building blocks to create two cropping-homomorphic signature schemes (Section 6).

We then describe how to convert any cropping-homomorphic signature scheme into a scaling-homomorphic signature scheme by observing that scaling an image is equivalent to cropping certain coefficients of its Discrete Cosine Transform (DCT) matrix[24] (Section 7). Thus, by representing the image as its DCT and signing the DCT representation with a croppable signature scheme, we can create an image format and accompanying signature that is homomorphic with respect to scaling. To support cropping and scaling simultaneously, we divide the image into blocks, take the DCT of each block, and then sign this data with a 4-dimensional cropping-homomorphic signature scheme. Cropping two of the four dimensions corresponds to deleting entire blocks, which is equivalent to cropping the original image. Cropping in the other two dimensions corresponds to cropping within all the DCT blocks, which corresponds to scaling each block of the original image, which is equivalent to scaling the original image. To enable lossy JPEG-like compression, we divide the DCT coefficients into their bitplanes and perform the above signatures on each bitplane independently. Thus, we represent the image as a 5-dimensional matrix in which cropping in each dimension corresponds to one of our supported image operations – cropping height, cropping width, scaling height, scaling width, and JPEG-like compression. We then describe a few tricks to simplify the signatures and make them more efficient.

Section 8 presents performance results for prototype implementations of our signature schemes. Our experiments show that, in the average case, our signatures can be substantially smaller than predicted by the worst-case analysis of Sections 4 and 5. We discuss open problems and make concluding remarks in Section 9.

2 Related Work

Multimedia Authentication Numerous authors have studied cryptographic methods of verifying the integrity of photographs and other digital media[13, 21, 3, 12, 34, 33, 32, 36, 35, 30, 7, 8, 26, 28, 27, 18], but these schemes all have at least of the following deficiencies: they are insecure, they are less efficient, or they support fewer image operations than our signature scheme. The JPEG2000 security extension has also inspired a substantial amount of research[18, 28, 8, 7, 30, 35]. Some of these schemes, particularly that of Peng, et al[18], use Merkle hash trees and thus may benefit by applying our multidimensional variant to reduce signature sizes or to support more image operations.

Statistical Forgery Detection Other researchers have developed statistical and other consistency tests to detect evidence of tampering in digital photographs[2, 4, 5, 14, 20, 19, 9, 17]. Unfortunately, all these tests are vulnerable to an "oracle" attack: an attacker can apply the same tests to his candidate image, grooming it until it passes. Thus these tests may catch a casual attempt at forging a photograph, but they cannot stop a determined fraudster.

Homomorphic Signatures Homomorphic signatures were first proposed by Rivest in 2001[22], and Micali and Rivest presented the first such scheme, for graphs, in 2002[15]. Johnson, et al, published their redactable and set-homomorphic signature schemes at the same conference[10]. These initial schemes have inspired others[25, 11, 1]. In contrast to these other schemes that develop signatures homomorphic with respect to one operation, this paper presents techniques for creating efficient signatures that are homomorphic with respect to several document operations simultaneously.

3 Redactable Signatures

Our signature schemes builds on the redactable signature scheme of Johnson, et al[10], so we summarize that scheme here.

Given a vector $x = (x_0, \ldots, x_{n-1})$, a vector x' is a redaction of x if x' is also of length n and for all $0 \le i < n$, $x'[i] = x[i]$ or $x'[i] = \perp$, where $\perp$ is a special symbol indicating the position i has been erased. A redactable signature scheme has three phases:

1. The private-key holder signs x, creating $s = \mathrm{Sig}(x)$, and transmits (x, s) to the redactor.
2. The redactor replaces some positions of x with $\perp$, creating a redacted vector x'. Simultaneously, the redactor uses s to derive a new signature s' on x'. Note that the redactor does not have access to the private signing key used to generate s. The redactor then publishes (x', s').
3. Some third party obtains (x', s') and uses the original signer's public key to verify that s' is a valid signature for x'.

As with normal signatures, redactable signatures should be unforgeable, but the notion of a forgery must be changed since anyone is allowed to construct signatures of redactions of signed vectors. A forger may also see several different signed redactions of the same original document. In general, given signed redactions $x_1, \ldots, x_n$ of some document x, let the *join*, x', of $x_1, \ldots, x_n$ be

$$x'[i] = \begin{cases} x_j[i] & \text{for some } j \text{ if } x_j[i] \neq \perp \\ \perp & \text{if } x_j[i] = \perp \text{ for all } j \end{cases}$$

The forger should not be able to construct a signature on some document x^* that is not a redaction of x'.

To model this scenario, we give the adversary acess to two oracles, S and R. The adversary can use oracle S to register vectors $x_1, \ldots, x_q$. For each registered vector, S computes and stores a signature s_i, but does not return it to the adversary. When the adversary makes a query $R(i, x)$ such that x is a redaction of x_i, the oracle uses s_i to compute a signature on x and returns it to the adversary. For each i, let W_i be the join of all vectors x that appear in a query of the form $R(i, x)$. We say the adversary has created an *existential forgery* if it produces a (possibly redacted) vector x^* with valid signature s^* such that x^* is not a redaction of W_i for any i.

Definition 1. *A redactable signature scheme is (t, q, ϵ)-secure against existential forgeries if, for all adversaries, A, running in time t and making at most q queries to S,*

$$Pr[A^{S,R} \text{ outputs an existential forgery}] \leq \epsilon$$

The redactable signature scheme of Johnson, et al, uses three building blocks: a length-doubling secure PRNG $G : \{0,1\}^{\ell} \to \{0,1\}^{2\ell}$, a hash function $H : \{0,1\}^* \to \{0,1\}^k$, and any standard signature algorithm, Sig_0. Let $G_0(r)$ and $G_1(r)$ be the first and second halves, respectively, of $G(r)$. To compute a redactable signature on a vector x of length n, the signer picks a random seed, k_ϵ, and executes the following steps.

1. Build a GGM tree of height $\log n$ from the seed k_ϵ. We can label each node of the tree according to the path from the root to the node. Thus, for example, a node k_w has children $k_{w0} = G_0(k_w)$ and $k_{w1} = G_1(k_w)$. Since the tree has height $\log n$, it contains one leaf for each entry in x, and we can interpret the label on each leaf as a binary integer to obtain the index of the corresponding element of x.
2. Set $v_w = H(0, k_w, x_w)$ for each entry in x, and build a Merkle hash tree from the v_w values via the rule $v_\eta = H(1, v_{\eta 0}, v_{\eta 1})$. The root of this tree will be v_ϵ.
3. Set $\mathrm{Sig}(x) = (k_\epsilon, \mathrm{Sig}_0(v_\epsilon))$.

Given a redactable signature $(k_\epsilon, \mathrm{Sig}_0(v_\epsilon))$ on some vector x, a verifier can use k_ϵ to repeat steps 1 and 2 of the above algorithm to obtain v_ϵ and then verify the signature $\mathrm{Sig}_0(v_\epsilon)$.

Suppose a redactor deletes a suffix of x to obtain x'. The recipient of x' will not be able to compute the values v_w that correspond to deleted entries in x', and hence will not be able to verify the signature. To overcome this problem, the redactor can reveal the hashes v_w corresponding to each deleted position in x'. Notice that the redactor can save space since, whenever he reveals two siblings $v_{\eta 0}$ and $v_{\eta 1}$, he could simply reveal the witness v_η instead. After recursively coalescing hashes in this way, the redactor only has to reveal the $O(\log n)$ hashes at the siblings of the nodes along the path from the right-most non-deleted entry in x' to the root.

If the entries of x are easy to guess, though, then an attacker who sees a redacted vector x' might be able to guess the missing entries and use the leaves of the GGM tree and the revealed hashes to verify his guesses. Thus, the redactor cannot let an attacker learn the GGM nodes corresponding to deleted positions of x'. To prevent this, the redactor erases k_ϵ from the signature and instead includes all the GGM leaves corresponding to non-deleted positions in x'. Then, as with the hash tree, the redactor can combine revealed siblings k_{w0} and k_{w1} and reveal only their parent, k_w.

Taken together, a redactor can delete the suffix of x and create a new signature on x' by revealing $O(\log n)$ GGM tree values and $O(\log n)$ hash witnesses. In general, deleting a contiguous region of x requires revealing $O(\log n)$ GGM nodes and $O(\log n)$ witnesses.

Given two signatures s_1 and s_2 on redactions x_1 and x_2, respectively, derived from a signature s of the original message x, one can create a signature s' on the vector x' defined by $x'[i] = x_1[i]$ if $x_1[i] \neq \perp$ and $x'[i] = x_2[i]$ otherwise. A position in x' is deleted only if it is deleted in both x_1 and x_2. This property seems harmless, since s' does not convey any new information that is not already apparent from s_1 and s_2. Our cropping-homomorphic signatures will have the same property.

3.1 A Naive 2-Dimensional Construction

The above construction efficiently supports redactions of contiguous sections of a signed vector x and we would like to extend this property to higher dimensional data. Specifically, we would

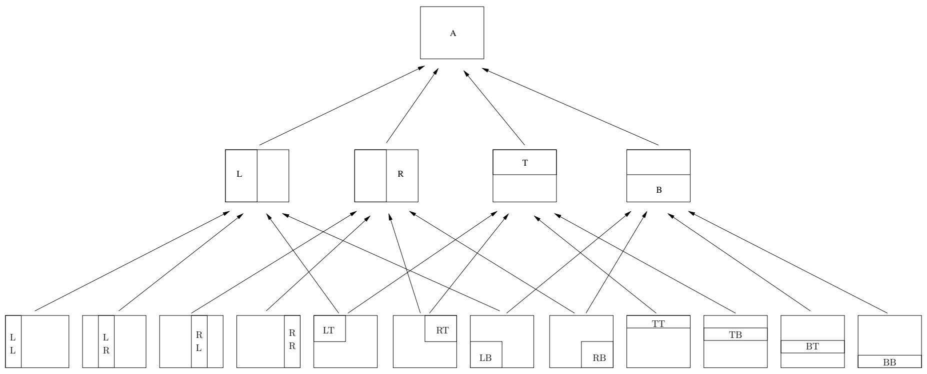


Fig. 1. Two levels of D_A, a 2-dimensional hash structure.

like a scheme for signing a matrix A such that we can efficiently "crop" the matrix, i.e. cut out a rectangular submatrix A', and produce a new signature for the cropped submatrix.

Naively applying the above scheme yields a workable but inefficient croppable signature scheme. Given an $H \times W$ matrix A, we can interpret A as a vector of length HW using row-major order and then sign this vector with the redactable scheme above. Using this arrangement, cropping an $h \times w$ submatrix from A corresponds to deleting $h+1$ contiguous regions from A's vector representation. Thus the signatures generated by this scheme contain $\Omega(h \log(W-w))$ witnesses and $\Omega(h \log w)$ GGM values. If we instantiate this scheme using 256-bit hashes and 128-bit PRNG seeds, then the signature on a cropped subimage of a 16 megapixel digital photograph can be over 2MB in size.

In this paper, we improve on this solution in two ways. First, we present a new hashing structure that reduces the number of hash witnesses in the signature to $O(\log HW \log hw)$. We then present two alternatives to the above GGM tree arrangement. The first alternative is provably secure against normal adversaries but only reduces the number of seeds in a signature to $O(w+h)$. The second construction is only provably secure against an adversary that makes 1 signing oracle query, but is probably secure given some reasonable assumptions about the entropy of pixels in digital photographs. With the second construction, we only need $O(\log hw)$ seeds in a cropped signature.

4 Merkle Hashing for Multi-Dimensional Data

Given a multi-dimensional array A, we can extend Merkle hashing to this array by defining a set of canonical subregions of the array, ordered by inclusion. These subregions will form a directed acyclic graph, and we can compute a hash for each subregion based on the hashes of its children. We begin by defining the subregions and hashing algorithm, and then we analyze the efficiency and security of this hashing scheme.

Let A be an $H \times W$ matrix. In the digital photograph application, A will be the pixels in a photograph. The matrix A induces a DAG D_A as follows. The reader may wish to refer to Figure 1, which shows two levels of D_A. A node $r \in D_A$ will be of the form $r = (Y, X)$ where X and Y are the intervals that determine the canonical subregion to which r corresponds. The DAG has a single root, $r_A = ([0, H-1], [0, W-1])$. A node $r = ([a,b],[c,d])$ has up to 4 children:

$$r^T = ([a, \lfloor \tfrac{a+b}{2} \rfloor], [c,d]) \qquad r^B = ([\lfloor \tfrac{a+b}{2} \rfloor + 1, b], [c,d])$$
$$r^L = ([a,b], [c, \lfloor \tfrac{c+d}{2} \rfloor]) \qquad r^R = ([a,b], [\lfloor \tfrac{c+d}{2} \rfloor + 1, d])$$

corresponding to r's top, bottom, left, and right halves, respectively. Every region has either 0, 2, or 4 children: if $a = b$ then r will not have top or bottom children; if $c = d$ then r will not have left and right children. Analogous to the notation from redactable signature schemes, nodes of D_A can be specified via strings from the language $\{R, L, T, B\}^*$. In this setting, the correspondence is not one-to-one because $r^{TR} = r^{RT}$ and $r^{BL} = r^{LB}$, although $r^{TB} \neq r^{BT}$ and $r^{LR} \neq r^{RL}$. If T_Y and T_X are binary interval trees on $[a, b]$ and $[c, d]$ respectively, then T_Y has $2H - 1$ nodes, T_X has $2H - 1$ nodes, and $D_A \equiv T_Y \times T_X$, so D_A has approximately $4HW$ nodes.

For each node of D_A, we compute a hash as follows. Let H_0 be a collision-resistant hash-function. Set

$$h_r = \begin{cases} H_0(0||A[a,b]) & \text{if } r = (\{a\}, \{b\}) \\ H_0(1||h_{r^T}||h_{r^B}) & \text{if } r = ([a,b], \{c\}), a \neq b \\ H_0(1||h_{r^R}||h_{r^L}) & \text{if } r = (\{a\}, [b,c]), b \neq c \\ H_0(1||h_{r^T}||h_{r^R}||h_{r^B}||h_{r^L}) & \text{otherwise} \end{cases}$$

Finally, set $H(A) = h_{r_A}$.

Security. Most security properties of Merkle hash trees carry over to multi-dimensional Merkle hashing because we can convert a Merkle DAG into a Merkle tree by "exploding" the DAG as follows. Let D_A be a Merkle DAG on an $H \times W$ matrix A. In this case, every leaf of D_A is at height $\log HW$ and each node has at most 4 children, so there are at most $4^{\log HW} = (HW)^2$ paths from the root to leaves. Let P be the set of paths in D_A, let $A[p]$ be the element of A at the end of a path p. Each path p has a corresponding word $w_p \in \{T, B, L, R\}^*$, and we can sort the paths according to the lexicographic ordering of their corresponding words. This induces a vector x of length at most $(HW)^2$, indexed by paths in P, and defined by $x[p] = A[p]$. The paths in P give rise to a tree structure over x, and performing Merkle hashing over this tree structure will yield the same result as in the original DAG over A.[2] Thus, for example, if an attacker can find a hash collision using multi-dimensional Merkle hashing, he can immediately convert this into a collision using standard Merkle hashing.

Efficiency. A memoized recursive algorithm, such as the 2d-hash procedure shown in Figure 2, can compute all these hashes in $O(HW)$ time because there are only $4HW$ hashes to compute and each hash can be computed from its children in constant time. In this procedure, C is the memoization cache of previously computed hashes, which we assume is passed by reference. The mappings M and O will be used to perform croppings but, for now, assume that they both have empty domains. The following theorem proves that the algorithm is also memory-efficient.

Theorem 1. *If domain(M) is empty, then, during the execution of 2d-hash on a region of size $H \times W$, the cache C never contains more than $4\min(H, W) + 3\max(H, W)$ elements.*

Proof. See companion technical report[6].

The algorithm can further reduce memory usage by not caching regions that span the original input range. These regions only have one parent in the DAG, so they do not need to be cached. Finally, we have implemented this algorithm and verified experimentally that it's performance is consistent with the theorem.

[2] Technically, this tree structure is not a Merkle tree since many of the internal nodes have 4 children. This doesn't affect the security of Merkle hashing, and we could eliminate this wrinkle by introducing suitable intermediate nodes in the DAG.

```
procedure 2d-hash ($A$, $r$, $s$, $C$, $M$, $O$)
  if $r \in$ domain($M$)
    $h := M[r]$
  else if $r \in$ domain($C$)
    $h := C[r]$
  else if $r = (\{a\}, \{c\})$
    $h := H_0(0||A[a, c])$
  else
    $t := \epsilon$
    $x := s$
    for i = 1, ..., 4
      if $r^x$ exists
        $t := t$ || 2d-hash($A$, $r^x$, $x$, $C$, $M$, $O$)
      $x :=$ next($x$)
    $h := H_0(1||t)$
  if $r \notin$ domain($C$)
    $C[r] := h$
  else
    delete $C[r]$
  if $r \in$ domain($O$)
    $O[r] := h$
  return $h$
```

```
procedure witness-set($R$, $\hat{r}$, $\hat{w}$, $r'$)
  if $r' = \emptyset$
    return $\{R\}$
  if $r' = R$
    return $\emptyset$
  if $r'$ spans $R$ in the $x$ or $y$ direction
    Pick $c, c' \in$ children($R$) such that $c$ and $c'$
          span $R$ in the same direction as $r'$
  else if $\hat{r} = R$
    Pick $c, c' \in_R$ children($R$) such that $c \cup c' = R$
  else
    Pick $\{c, c'\}$ = children($R$) $\setminus \hat{w}$
  return (children($R$) $\setminus \{c, c'\}$) $\cup$
         witness-set($c$, $c \cap \hat{r}$, $\hat{w}$, $c \cap r'$) $\cup$
         witness-set($c'$, $c' \cap \hat{r}$, $\hat{w}$, $c' \cap r'$)

procedure next(T) = R
          next(R) = B
          next(B) = L
          next(L) = T
```

Fig. 2. The 2d-hash and witness-set algorithms. Note that, for 2d-hash, C and O are passed by reference. The "next" procedure just defines a clockwise ordering of T, R, B, and L.

Witness Sets. Suppose that A' is a submatrix of A covering subregion $r' \subseteq r_A$. Here, r' can be any subregion of r_A, not just the regions present in D_A. We can construct a proof that A' was part of the data used to compute h_{r_A} by revealing the hash values at appropriate nodes in D_A. For example, if A' is contained entirely in the bottom half of A, then we can reveal the hashes $h_{r_A^T}$, $h_{r_A^R}$, and $h_{r_A^L}$. This then reduces the problem to proving that A' was part of the data used to compute $h_{r_A^B}$. The witness-set procedure in Figure 2 computes the set of nodes whose hashes must be revealed to prove that the data in some region r' was used in the hash computation for data in region r. The following theorem bounds the size of the set of revealed witnesses.

Theorem 2. *Suppose region r is an $H \times W$ rectangle, region $r' \subseteq r$ is an $h \times w$ rectangle, and O =witness-set(r, r'). Then $|O| \leq 12 \log HW \log hw$.*

Proof. See companion technical report[6].

The bound in Theorem 2 is rather conservative, and our experiments in Section 8 show that, on average, this theorem over-estimates the number of witnesses by a factor of 3. Furthermore, we can make witness sets even smaller by introducing intermediate nodes in the hash DAG. For example, if we change the definition of the hash to

$$h_r = \begin{cases} H_0(0||A[a,b]) & \text{if } r = (\{a\},\{b\}) \\ H_0(1||h_{r^T}||h_{r^B}) & \text{if } r = ([a,b],\{c\}), a \neq b \\ H_0(1||h_{r^R}||h_{r^L}) & \text{if } r = (\{a\},[b,c]), b \neq c \\ H_0(1||H_1(h_{r^T}||h_{r^R})||H_1(h_{r^B}||h_{r^L})) & \text{otherwise} \end{cases}$$

then witness-set only has to give away one or two witnesses at each step instead of two or three. Overall, this reduces the constants in the above analysis to $8 \log HW \log hw$.

Theorem 2 assumes that the submatrix A' is rectangular, but it is possible to construct a set of witnesses for a subregion of any shape. We have not attempted to analyze the size of the witness sets that would be required.

It may sometimes be desirable to take a witness set for a submatrix A' of A and construct another witness set for a submatrix A'' of A'. In the realm of digital photographs, this would correspond to cropping an already cropped photograph. The next theorem confirms that it is always possible to construct a small witness-set for A'' from A' and its witness set.

Theorem 3. *Let A'', A', and A be matrices covering regions $r'' \subseteq r' \subseteq r$, respectively. For every witness set $O =$witness-set(r,r'), there exists a witness set $O' =$witness-set(r,r'') such that all the witnesses in O' can be computed from A' and the witnesses specified by O.*

Proof. See companion technical report[6].

Since O' is selected by the witness-set procedure, it must be small, as established in Theorem 2.

5 PRNGs for Croppable Signatures

In this section we describe two methods for generating a matrix of random mask values for use in a multi-dimensional croppable signature scheme. Recall that the method from Section 3 built a standard 1-dimensional GGM tree and then organized its leaves into a matrix using row-major order. When used in a croppable signature scheme, the signature for an $h \times w$ submatrix of a signed $H \times W$ matrix must include $\Omega(h \log w)$ values from the GGM tree. Our first improvement uses the same technique with a better space-filling curve to achieve $O(h + w)$ tree nodes in a signature. Since it is merely a re-arrangement of the outputs of the GGM tree, it is clearly just as secure as the original scheme. Our second method sacrifices some security to substantially reduce the number of seed values that must be included in a signature. With our second scheme, signatures need to include only $O(\log hw)$ seeds.

Morton-order PRNG. Let r be an $H \times W$ region, and let r^T, r^B, r^L, and r^R be the top, bottom, left, and right halves of r, as in the previous section. We label nodes in the GGM tree with strings from the language $\{T, B, L, R\}^*$. Given a random seed k_ϵ, we generate GGM values recursively using the formula

$$G(k_w) = \begin{cases} (k_{wL}, k_{wR}) & \text{if } r^w \text{ is wider than it is tall} \\ (k_{wT}, k_{wB}) & \text{otherwise} \end{cases}$$

Note that if r^w corresponds to a region of size 1, then k_w is a leaf of this tree. Each leaf value k_w is mapped to location r^w in the final output matrix. Since this algorithm is simply a binary GGM tree, it generates the same values as the original solution in Section 3, but arranges them in the final output matrix using a variant of the Morton-order space-filling curve[16]. The following well-known theorem, originally in the context of quad-trees[23], establishes the $O(h + w)$ bound promised above.

Theorem 4. *Let r be an $H \times W$ region, let R be the set of regions corresponding to nodes in the Morton-order GGM tree defined above, and let $r' \subseteq r$ be any $h \times w$ subregion of r. There exists a set of at most $4(h + w)$ disjoint regions $r_1, \ldots, r_m \in R$ such that $r' = \bigcup_{i=1}^m r_i$.*

Unfortunately, $4(w + h)$ can be quite large for digital photograph applications. For example, a subimage of a 16MP photograph could require about 2^{15} GGM tree values. Using 16-byte seeds, this would create a signature over $500KB$ in size.

Intersecting PRNGs. Our alternative solution is much more efficient, but sacrifices security. Pick a random seed k_ϵ, set $(x_\epsilon, y_\epsilon) = G(k_\epsilon)$, and use normal GGM trees to generate $x_0, \ldots, x_{W-1}$ and $y_0, \ldots, y_{H-1}$. Output i, j is simply $x_i || y_j$. This construction is not a PRNG, but it is now easy to reveal the outputs in any region r': reveal the x outputs spanning r' horizontally and the y outputs spanning r' vertically. All total, this only reveals $\log w + \log h = \log hw$ tree nodes. The price of this efficiency is that an adversary that obtains the outputs of this generator in two different regions, $r_1 = ([a_1, b_1], [c_1, d_1])$ and $r_2 = ([a_2, b_2], [c_2, d_2])$, can combine the seeds to learn the outputs on two other regions: $([a_1, b_1], [c_2, d_2])$ and $([a_2, b_2], [c_1, d_1])$. Nonetheless, we can use this scheme to build a croppable signature scheme that is secure against an adversary that only makes one query to the signing oracle, as proven in the next section.

6 Cropping-Homomorphic Signatures

We can now build two different cropping-homomorphic signature schemes.

Morton-curve-based signature scheme. To sign a matrix A corresponding to region r, the signer picks a random seed k_ϵ and executes the following algorithm:

1. Use a GGM tree and Morton-curve to generate a matrix of pseudo-random values, $k[i, j]$.
2. Set $v[i, j] = H(0, k[i, j], A[i, j])$ and use multi-dimensional Merkle hashing to compute a hash, v_ϵ of the matrix v.
3. Output $\mathrm{Sig}_M(A) = (r, k_\epsilon, \mathrm{Sig}_0(v_\epsilon))$.

As explained in previous sections, a cropper can construct a signature on a submatrix A' of size $h \times w$ by deleting k_ϵ from the signature of A and including $4(w + h)$ GGM values and $O(\log HW + (\log hw)^2)$ hash values. The signature should also specify the location, r', of A' within A. Thus, the format of a cropped signature is $(r, r', \{k_w\}, \{v_w\}, \mathrm{Sig}_0(v_\epsilon))$.

The definition of security for cropping-homomorphic signatures is directly analogous to the definition of security for redactable signatures. The next theorem is analogous to the security theorem for redactable signatures.

Theorem 5. *Let $G : \{0,1\}^m \to \{0,1\}^{2m}$ be a (t, ϵ_G)-secure PRNG, H a (t, p_H)-collision-resistant hash function, and Sig_0 a signature scheme (t, q, p_S)-secure against existential forgeries. Suppose also that $H(0, k, \cdot)$ is a (t, q, ϵ_H)-secure PRF (indexed by k. Then the cropping-homomorphic signature scheme Sig_M defined above, when used to sign matrices of size at most $H \times W$, is (t', q, p')-secure against existential forgeries, where $t' \approx t$ and $p' = p_S + p_H + qHW\epsilon_G + qHW\epsilon_H + qHW2^{-m}$.*

Proof. See companion technical report[6].

Intersection-based signature scheme. To reduce the number of PRNG seeds that must be revealed when cropping a matrix, we can replace the Morton-curve construction with the PRNG intersection scheme presented in Section 5, yielding a new signature scheme which we call Sig_I. This scheme is otherwise identical to the one above. This change reduces the number of seeds in a cropped signature from $O(h + w)$ to $O(\log hw)$, but reduces security, as the following theorem makes explicit.

Theorem 6. *Let G and Sig_0 be as in Theorem 5 and assume that H is a (t, p_H) collision-resistant hash function. Suppose also that $H(0, x||y, \cdot)$ is a (t, q, ϵ_H) PRF indexed by x and a (t, q, ϵ_H) PRF* *indexed by y. In other words, if the attacker gets to choose one of x and y and the other is chosen randomly, the attacker cannot distinguish $H(0, x||y, \cdot)$ from a random function. Let $\mathcal{A}$ be an adversary that makes at most one query $S(i, \cdot)$ for each i. Then the probability that $\mathcal{A}$ successfully constructs an existential forgery against Sig_I is at most $p_S + p_H + q(H+W)\epsilon_G + qHW\epsilon_H + qHW2^{-m}$.*

Proof. See companion technical report[6].

7 Other Homomorphic Signatures for Photographs

The croppable signature scheme above is already useful in the context of digital photographs, but we can use it as a foundation for building other homomorphic signature schemes.

Scaling-homomorphic signatures. Scaling and cropping are connected via the Discrete Cosine Transform (DCT)[24], so we can use this to immediately convert any cropping-homomorphic signature scheme into a scaling-homomorphic scheme. These two operations are related by the equation

$$\mathrm{Scale}_{h\times w}^{H\times W} = \mathrm{Clamp}_0^{255} \circ \sqrt{\frac{hw}{HW}} \circ \mathrm{DCT}^{-1} \circ \mathrm{Crop}_{h\times w} \circ \mathrm{DCT}$$

where $\mathrm{Scale}_{h\times w}^{H\times W}$ scales an $H \times W$ image down to an $h \times w$ image, Clamp constrains values to be in the range of pixel values (typically 0 to 255), $\sqrt{\frac{hw}{HW}}$ is a normalization factor, $\mathrm{Crop}_{h\times w}$ crops a matrix to only include its upper left $h \times w$ submatrix, and DCT is the Discrete Cosine Transform. Thus, for any cropping-homomorphic signature scheme Sig_C, $\mathrm{Sig}_S(I) = \mathrm{Sig}_C(\mathrm{DCT}(I))$ is a scaling-homomorphic signature scheme.

The Clamp operation and $\sqrt{\frac{hw}{HW}}$ normalization introduce a slight wrinkle in the scheme. After scaling I to I' using the above algorithm, we can construct a new signature s' on $\mathrm{Crop}(\mathrm{DCT}(I))$, so a signature verifier must be able to compute $\mathrm{Crop}(\mathrm{DCT}(I))$ in order to verify the signature. Since the Clamp operator and the rounding performed in the normalization by $\sqrt{\frac{hw}{HW}}$ are both non-invertible, it is not possible to reconstruct $\mathrm{Crop}(\mathrm{DCT}(I))$ from $I' = \mathrm{Clamp} \circ \sqrt{\frac{hw}{HW}} \circ \mathrm{DCT}^{-1} \circ \mathrm{Crop} \circ \mathrm{DCT}(I)$. Thus, after scaling I, we must store and transmit I' as $D' = \mathrm{Crop}(\mathrm{DCT}(I))$. The final recipient of an image will have to compute $I' = \mathrm{Clamp} \circ \sqrt{\frac{hw}{HW}} \circ \mathrm{DCT}^{-1}(D')$ before displaying it.

Scaling- and cropping-homomorphic signatures. Let Sig_C be a 4-dimensional croppable signature scheme. Although we have only presented 2-dimensional constructions in this paper, they all generalize easily to higher-dimensions, so we know that such a signature scheme exists under standard cryptographic assumptions. To sign an $H \times W$ image I, divide I into $B_1 \times B_2$ blocks, each of size $\frac{H}{B_1} \times \frac{W}{B_2}$, creating a 4-dimensional array $B[i, j, k, \ell]$, where indices i and j select a block and k and ℓ select a pixel within that block. Compute the DCT of each block separately, creating a new array $D[i, j] = \mathrm{DCT}(B[i, j])$. Let $\mathrm{Sig}_{CS}(I) = \mathrm{Sig}_C(D)$. As with the scaling-homomorphic scheme, we must store and transmit D instead of the original image.

Now consider the different cropping operations we can perform on D. Cropping in the i or j dimensions crops entire rows or columns of blocks of D, which corresponds to cropping entire rows or columns of blocks of B, which corresponds to cropping the original image I by a multiple of the block size. Cropping D in the k or ℓ dimension scales each block $B[i, j]$ by the same amount, and scaling each block separately is equivalent to scaling the image by a multiple of the number of blocks. Thus, with some granularity, we can scale and crop the original image while preserving the signature. We recommend a block size of $\sqrt{H} \times \sqrt{W}$. For typical digital photographs, this would yield blocks of size about 32×32 or 64×64.

Curiously, cropping the image actually *improves* the granularity of subsequent scaling operations, and vice versa. To see why, consider cropping the image down to single block. We can now scale the size of that block with perfect granularity. Conversely, if we scale the image down so that each block contains exactly a single pixel, then we can crop to any desired size.

We can simplify this signature scheme by sacrificing some scaling power. Suppose we wish to support only aspect-ratio preserving scalings. In this case, each (square) block of D can only be cropped to a square sub-block, so we can re-order the coefficients in each block into a one-dimensional array, with the last-to-be-cropped coefficients at the beginning of the array and the first-to-be-cropped coefficients at the end of the array. This reduces D to a 3-dimensional array, so we can build such a cropping- and scaling-homomorphic signature scheme from a 3-dimensional cropping-homomorphic scheme.

JPEG compresses images by dividing them into 8×8 pixel blocks, computing the DCT on each block, and quantizing the DCT coefficients. Thus the scaling and cropping homomorphic signature scheme would mesh well with a generalized version of JPEG that supports arbitrary block sizes.

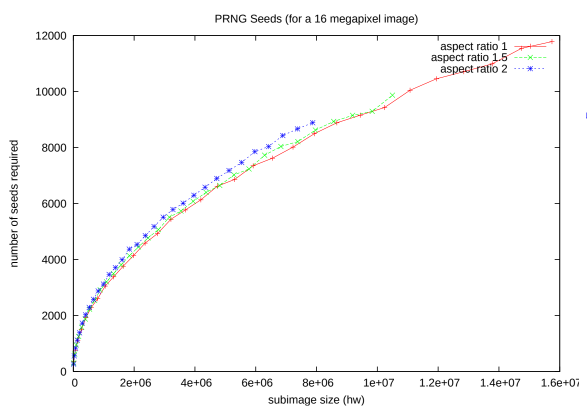
JPEG-like compression. JPEG applies a block-by-block DCT transform, as we do in the scalable and croppable scheme above, although JPEG always uses 8×8 blocks. The only lossy step performed in JPEG compression is "quantization", in which the coefficients in each block are divided by a constant to reduce the number of bits required to represent them, although with a corresponding loss of precision. Note that JPEG enables different quantization factors for each coefficient.

We can support a limited form of quantization quite simply by dividing the coefficients in D into their individual bits, turning D into a 5-dimensional array $D[i, j, k, \ell, b]$, where index b selects the desired bit of coefficient k, ℓ of block i, j. Cropping away the c least-significant bits quantizes all the coefficients by a factor of 2^c. Compressing different coefficients by different quantization factors, a feature crucial to good JPEG compression, is possible with the croppable signature schemes of Section 6, but the resulting signatures will be larger. Even with this extension, though, quantization factors must be powers of 2.

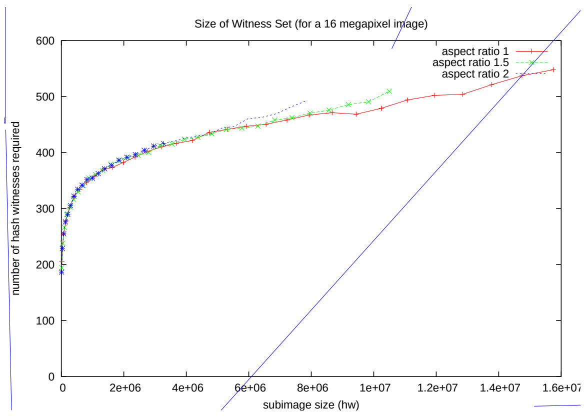
8 Experimental Results

Figure 3 plots the average seed set size and hash witness set size for a subimage cropped from a 4096×4096 digital photograph versus the number of pixels in the subimage. These figures are based on a 2-dimensional hashing scheme that only supports cropping. For a randomly selected $h \times w$ subimage, the expected seed set size is approximately $3\sqrt{hw}$. The size of witness sets is much more efficient than predicted, averaging less than $4(\log hw)^2$ instead of the predicted $12 \log HW \log hw$. In practice, this means that the size of a signature will be dominated by the size of the seed set. Signature size is also independent of the aspect ratio of the cropped subimage.

Using 256-bit hash values and 128-bit PRNG seeds, the largest signature we expect to see will be around 210KB, which only occurs when the cropped subimage is 15 megapixels. In this case, the JPEG representation of the subimage would be over 2MB, giving a signature overhead of 10%.



(a)



(b)

Fig. 3. Average (a) witness set and (b) seed set sizes for subimages of different sizes, all from 16MP digital photographs.

Our hashing scheme requires computing many small hashes. On a 2.8GHz Pentium, the OpenSSL implementation of SHA-256 can perform about 275,000 hashes per second. Hashing a 16MP image requires 64 million hash operations, which would take just under 4 minutes. A digital camera implementing our signature scheme could include specialized hardware for performing hashes, and could also perform them offline while otherwise idle.

9 Conclusion

We have presented new hashing and PRNG constructions for building efficient signature schemes that are homomorphic with respect to cropping of multi-dimensional data. We then presented several useful applications to authenticating digital photographs and other media, including signature schemes that are simultaneously homomorphic with respect to cropping, scaling, and JPEG-like compression. This research leaves several open questions. Is there a provably secure matrix PRNG construction that admits small seed-sets in croppable signatures? If we had four one-way functions, f_T, f_B, f_R, and f_L such that f_T and f_B commute with f_R and f_L, but not with eachother, then we could construct such a PRNG by letting $A_w = f_{w_1}(f_{w_2}(\dots(f_{w_n}(r))))$. Commuting one-way functions may exist, for example, two instances of the RSA function with the same modulus but different exponents will commute, but we have not found a secure, working scheme based on this idea. Is there a generic way to construct signature schemes that are simultaneously homomorphic with respect to two different document operations? For digital photographs, is there a croppable and scalable signature scheme with perfect granularity? What about other important, generally benign image operations, such as sharpen, brighten, and contrast adjustments?

References

1. G. Ateniese, D.H. Chou, B. de Medeiros, and G. Tsudik. Sanitizable signatures. In *Proceedings of the 10th European Symposium on Research in Computer Security*, 2005.
2. Wen Chen, Yun Q. Shi, and Guorong Xuan. Identifying computer graphics using hsv color model and statistical moments of characteristic functions. In *Proceedings of the 2007 IEEE International Conference on Multimedia*.

3. Andrew Daviel. Trusted digital camera. `http://andrew.triumf.ca/trustcam`; viewed 2/6/2006.
4. H. Farid. Detecting digital forgeries using bispectral analysis. Technical report, MIT, 1999.
5. H. Farid and S. Lyu. Higher-order wavelet statistics and their application to digital forensics. In *Proceedings of the 2003 IEEE Workshop on Statistical Analysis in Computer Vision*.
6. Anonymized for submission. Homomorphic signatures for digital photographs. Technical report, Anonymized for submission, 2010.
7. Raphael Grosbois, Pierre Gerbelot, and Touradj Ebrahimi. Authentication and access control in the jpeg 2000 compressed domain. In *Proceedings of the 2001 SPIE Meeting no Applications of Digital Image Processing*.
8. Ayman Haggag, Takashi Yahagi, and Jianming Lu. Image authentication and integrity verification using jpsec protection tools. In *Proceedings of the First International Workshop on Image Media Quality and its Applications*, 2005.
9. M. Johnson and H. Farid. Exposing digital forgeries by detecting inconsistencies in lighting. In *Proceedings of the 2005 ACM Multimedia and Security Workshop*.
10. Robert Johnson, David Molnar, Dawn Song, and David Wagner. Homomorphic signature schemes. In *RSA Cryptographer's Track*, 2002.
11. Eike Kiltz, Anton Mityagin, Saurabh Panjwani, and Barath Raghavan. Append-only signatures. In *Automata, Languages and Programming*, 2005.
12. Ching-Yung Lin and Shih-Fu Chang. Generating robust digital signature for image/video authentication. In *Multimedia and Security Workshop at ACM Multimedia*, Bristol, UK, September 1998. http://www.ctr.columbia.edu/ cylin/pub/acmmm98.ps.
13. C.Y. Lin and S.F. Chang. Generating robust digital signature for image/video authentication. In *Proceedings of Multimedia and Security Workshop at ACM Multimedia*, September 1998.
14. J. Lukas, J. Fridrich, and M. Goljan. Detecting digital image forgeries using sensor pattern noise. In *Proceedings of the 2006 SPIE Meeting no Electronic Imaging and Photonics West*.
15. Silvio Micali and Ron Rivest. Transitive signature schemes. In *Proceedings of the 2002 RSA Conference*.
16. G.M. Morton. A computer oriented geodetic data base; and a new technique in file sequencing. Technical report, IBM, Ottawa, Canada, 1966.
17. T. T. Ng, S. F. Chang, J. Hsu, L. Xie, and M. P. Tsui. Physics-motivated features for distinguishing photographic images and computer graphics. In *Proceedings of the 2005 ACM Multimedia Conference*.
18. C. Peng, R. Deng, Y. Wu, and W. Shao. A flexible and scalable authentication scheme for jpeg2000 codestreams. *ACM Multimedia*, pages 433—441, November 2003.
19. A. Popescu and H. Farid. Statistical tools for digital forensics. In *Proceedings of the 2004 International Workshop on Information Hiding*.
20. A. Popescu and H. Farid. Exposing digital forgeries by detecting traces of re-sampling. *IEEE Transactions on Signal Processing*, 55(2):758—767, 2005.
21. CNET Reviews. Canon EOS-1Ds Digital SLR. `http://reviews.cnet.com/Canon_EOS_1Ds_Digital_SLR/4514-6501_7-20610303.html`; viewed 2/6/2006., February 2006.
22. Ron Rivest. Two new signature schemes. Cambridge Seminar Series, 2001.
23. Hanan Samet. *The Design and Analysis of Spatial Data Structures*. Addison-Wesley, 1989.
24. A.N. Skodras and C.A. Christopoulos. Down-sampling of compressed images in the dct domain. In *Proceedings of the 8th European Signal Processing Conference*, September 1998.
25. Ron Steinfeld, Laurence Bull, and Yuliang Zheng. Content extraction signatures. In *Information Security and Cryptology*, volume 2288, pages 163—205, 2001.
26. Q. Sun and S.-F. Chang. A secure and robust digital signature scheme for jpeg2000 image authentication. *IEEE Transactions on Multimedia*, 7(3):480—494, June 2005.
27. Q. Sun, S. F. Chang, M. Kurato, and M. Suto. A crypto signature scheme for image authentication over wireless channel. In *Proceedings of the 2004 IEEE International Conference on Multimedia*.
28. Q. Sun, S. F. Chang, M. Kurato, and M. Suto. A quantitative semi-fragile jpeg2000 image authentication system. In *Proceedings of the 2002 International Conference on Image Processing*.
29. Nicholas Wade. Journal to examine how it reviewed articles. New York Times, January 2006.
30. Susie Wee and John Apostolopoulos. Secure transcoding with jpsec confidentiality and authentication. Technical report, Hewlett-Packard, 2004.
31. Donald R. Winslow. Reuters apologizes over altered lebanon war photos; suspends photographer. National Press Photographers Association, August 2006.
32. J. Wu, B. B. Zhu, S. Li, and F. Lin. A secure image authentication algorithm with pixel-level tamper localization. In *Proceedings of the 2004 IEEE International Conference on Image Processing*.

33. J. Wu, B. B. Zhu, S. Li, and F. Lin. New attacks on sari image authentication system. In *Proceedings of the 6th SPIE Security, Steganography, and Watermarking of Multimedia Conference*, 2004.
34. Hong Heather Yu. Scalable multimedia authentication. In *Proceedings of the Fourth Pacific Rim Conference on Multimedia*, December 2003.
35. Zhishou Zhang, Gang Qiu, Qibin Sun, Xiao Lin, Zhicheng Ni, and Y.Q. Shi. A unified authentication framework for jpeg2000. In *Proceedings of the 2004 IEEE International Conference on Multimedia*.
36. B. B. Zhu, M. D. Swanson, and S. Li. Encryption and authentication for scalable multimedia: Current state of the art and challenges. In *Proceedings of the 5th SPIE Internet Multimedia Management Systems Conference*, 2004.