Trade-Off Between Key Size and Efficiency in Universal Hashing Using Polynomials

Palash Sarkar

Applied Statistics Unit
Indian Statistical Institute
203, B.T. Road, Kolkata
India 700108.
email: palash@isical.ac.in

Abstract. Consider the following universal hashing set-up. Let $\mathbb{F}$ be a finite field and suppose any message consists of at most 2^{30} elements of $\mathbb{F}$. Suppose further that a collision bound of 2^{-128} is desired. This can be achieved using usual polynomial hashing with $|\mathbb{F}| \approx 2^{160}$ and having a digest of 160 bits and a key of length also 160 bits; hashing an n-bit message requires approximately $n/160$ multiplications over $\mathbb{F}$.
We describe a new hashing method which can achieve the same collision bound of 2^{-128} for messages of at most L elements by working over a field $\mathbb{F}$ of size $\approx 2^{136}$. Hashing an n-bit message requires approximately $n/136$ multiplications over $\mathbb{F}$. Supposing similar efficiency in implementation of field arithmetic in both cases, the ratio of the two processing times is $160/136 \times [M_{136}]/[M_{160}]$, where M_a is the time for one multiplication in a field of size $\approx 2^a$. Since M_a is quadratic in a, the new algorithm is expected to be faster than usual polynomial hashing. Further, the size of the digest also reduces to 136 bits. The trade-off is that the key size increases to $5 \times 136 = 680$ bits compared to the key size of 160 bits for usual polynomial hashing.
The method mentioned above is a special case of a general method of combining independent universal hash functions at multiple levels to obtain a new universal hash function. Other consequences of the new algorithm are worked out including how it can be instantiated by a class of polynomials introduced by Bernstein.
Keywords: universal hash function.

1 Introduction

Universal hash functions are of fundamental importance in cryptography and computer science [13, 5]. Being of practical importance, there have been efforts [7, 11, 4, 1] to design hash functions which are very fast.

Polynomial based universal hashing over a finite field $\mathbb{F}$ has a collision bound of $(L-1)/|\mathbb{F}|$ for hashing messages consisting of at most L elements of $\mathbb{F}$. Both the key and the digest is a single element of $\mathbb{F}$. If a collision bound of $2^{-\rho}$ is desired, then the size of $\mathbb{F}$ has to be $\approx L2^{\rho}$ and so both the key and the digest are of size $\approx \rho + \log_2 L$. Hashing an n-bit message requires approximately $n/(\rho + \lceil \log_2 L \rceil)$ multiplications over $\mathbb{F}$. Bernstein [2] defined a new class of polynomials for which the number of multiplications can be halved at the negligible cost of increasing the collision bound to $(2L-1)/|\mathbb{F}|$.

The motivation for our work is to investigate whether the collision bound can be reduced. For certain hash functions [6, 5], the collision bound can be made $1/|\mathbb{F}|$ at the cost of making the key length equal to that of the message. The question that we ask is whether there is some intermediate trade-off.

To this end, we first propose a general construction of universal hash function. This construction builds a hash function working on arbitrary length strings from hash functions working on fixed size domains. In this sense, the new construction can be considered to be a domain extender for universal hash function. The basic idea is to take mutually independent hash functions $f_1, \ldots, f_\ell$ and combine them using a multi-level construction which uses one hash function at each level. Hashing messages with variable lengths requires an additional subtlety which is taken care of by using another XOR universal random function ψ.

Instantiating the functions f_i using usual polynomial hashing gives rise to a multi-variate polynomial. The collision bound is $m\lceil\log_m L\rceil/|\mathbb{F}|$, where L is as before the maximum number of elements of $\mathbb{F}$ in any message and m can be any value ≥ 2. To achieve a collision bound of $2^{-\rho}$, we need $|\mathbb{F}| \approx m\log_m L 2^\rho$. For fixed values of L and ρ and for an appropriately chosen m, the size of the field in the new algorithm is smaller than the size of the field required for polynomial hashing. Hashing an n-bit message requires approximately $n/(\rho+\log_2 m+\log_2(\log_m L))$ multiplications over $\mathbb{F}$. So the new algorithm requires more multiplications over a smaller field. The ratio of the time required by the new algorithm to time required for polynomial hashing is approximately $a/b \times [M_b]/[M_a]$, where $a = \rho + \lceil\log_2 L\rceil$ and $b = \rho + \log_2 m + \log_2(\log_m L)$ with $b < a$ for a suitably chosen m. Asymptotically, M_k is quadratic in k and hence the new algorithm will require less time. For concrete values of k, however, M_k may not have a strict quadratic dependence on k. To a certain extent, this will be determined by the implementation at hand, though, we expect the decrease in the size of the field to have a more significant impact on the run-time than the increase in the number of multiplications.

Further, the digest in the new algorithm is a single element of $\mathbb{F}$ and hence is of size b bits. The trade-off is that the size of key is increased to approximately $b(\log_m L + 1)$ bits. Putting $L = 2^{30}$, $\rho = 128$ and choosing $m = 2^6$ justifies the values given in the abstract.

The basic multi-level construction can also be combined with the class of polynomials due to Bernstein [2] based on earlier work due to Rabin and Winograd [8], which we call the BRW polynomials. More interestingly, it is possible to instantiate some of the f_i's using BRW polynomials and the others using usual polynomials. The cost analysis of the multi-level hash function when the f_i's are BRW polynomials and a possible application for combining BRW and usual polynomial hashing is dicussed later.

Prior and related works. Universal hash functions were introduced in [5] and have been extensively studied since then [13, 3, 12, 9]. To the best of our knowledge, the construction that we describe here has not been reported earlier.

2 Preliminaries

A random (but, not necessarily uniform random) function $f : \{0,1\}^* \rightarrow \{0,1\}^t$ is said to have a collision bound of ε, if for distinct $x, x' \in \{0,1\}^*$, $\Pr[f(x) = f(x')] \leq \varepsilon$. Such a function is said to be ε-almost universal (ε-AU). The function f is said to be ε-almost XOR universal (ε-AXU) if for distinct $x, x' \in \{0,1\}^*$ and any $\alpha \in \{0,1\}^t$, $\Pr[f(x) \oplus f(x') = \alpha] \leq \varepsilon$. A weaker notion of AU is to require that the collision bound holds only if the distinct strings x, x' have equal lengths. Let $\mathbb{F}$ be a finite field. Then a random function $f : \cup_{i=1}^{r}\mathbb{F}^i \rightarrow \mathbb{F}$ is defined to be ε-AU or ε-AXU in a manner similar to above.

Elements in the domain of f are called messages and the value $f(x)$ is called the digest of x obtained using f. A random function f is realised from a function family $\{f_K\}_{K\in\mathcal{K}}$, where $\mathcal{K}$ is called the key space. A key K is chosen uniformly at random from $\mathcal{K}$ and then f is set to f_K. The randomness in f comes from the random choice of K and hence, the probabilities mentioned above are over the random choices of K.

For $(x_1,\ldots,x_r)\in\mathbb{F}^n$ define $\mathsf{poly}_\alpha(x_1,\ldots,x_r)=x_r+x_{r-1}\alpha+\cdots+x_r\alpha^{r-1}$. Using Horner's rule $\mathsf{poly}_\alpha(x_r,\ldots,x_1)$ can be evaluated using $(r-1)$ multiplications (over $\mathbb{F}$) and can be shown to be $(r-1)/|\mathbb{F}|$-AU for equal length messages. Further, $\alpha\mathsf{poly}_\alpha(x_1,\ldots,x_r)$ is $r/|\mathbb{F}|$-AXU for equal length messages and can be evaluated using r multiplications.

Bernstein [2] has introduced a class of polynomials which builds upon previous work due to Rabin and Winograd [8]. We call these the BRW polynomials. For $n\geq 2$, $\mathsf{BRW}_\alpha(x_1,\ldots,x_r)$ is $(2r-1)/|\mathbb{F}|$-AU and can be evaluated using $\lfloor r/2\rfloor$ multiplications and $\lg r$ squarings (to compute the powers $\alpha^2,\alpha^4,\ldots$). Further, $\alpha\mathsf{BRW}_\alpha(x_1,\ldots,x_r)$ is $2r/|\mathbb{F}|$-AXU.

We will need another kind of AXU function. A random function $\psi:\{1,\ldots,r\}\rightarrow GF(2^t)$ is said to be $1/2^t$-AXU if for distinct i,j with $1\leq i,j\leq r$ and any $\alpha\in GF(2^t)$, $\Pr[\psi(i)\oplus\psi(j)=\alpha]=1/2^t$.

Let $\tau(x)$ be a primitive polynomial of degree t over $GF(2)$ and suppose that this $\tau(x)$ is used to define $GF(2^t)$. Let α be a uniform random element of $GF(2^t)$ and define $\psi:i\mapsto x^i\alpha \bmod \tau(x)$. The element α is the key to the function ψ. It is not difficult to show that such a ψ satisfies $\frac{1}{2^t}$-AXU property if $r\leq 2^t-2$. A general definition of ψ and a more efficient instantiation using word oriented LFSRs can be found in [10].

3 The New Construction

The basic idea behind the construction is given in the following result.

Lemma 1. *Let $\mathbb{F}$ be a finite field and for $i=1,2$, $f_i:\mathbb{F}^{m_i}\rightarrow\mathbb{F}$ be independent random functions having collision bounds ε_i. Define a random function $f:\mathbb{F}^{m_1m_2}\rightarrow\mathbb{F}$ by*

$$f(\mathbf{x}_1,\ldots,\mathbf{x}_{m_2})=f_2(f_1(\mathbf{x}_1),\ldots,f_1(\mathbf{x}_{m_2})) \tag{1}$$

where each $\mathbf{x}_j\in\mathbb{F}^{m_1}$. Then a collision bound for f is $\varepsilon_1+\varepsilon_2$.

Proof: Let $\mathbf{x}=(\mathbf{x}_1,\ldots,\mathbf{x}_{m_2})$ and $\mathbf{x}'=(\mathbf{x}'_1,\ldots,\mathbf{x}'_{m_2})$ be two distinct elements of $\mathbb{F}^{m_1m_2}$. Define $y_i=f_1(\mathbf{x}_i)$, $y'_i=f_1(\mathbf{x}'_i)$ and let Eq be the event $\wedge_{i=1}^{m_2}(y_i=y'_i)$.

$$\begin{aligned}
\Pr[f(\mathbf{x})=f(\mathbf{x}')] &= \Pr[f(\mathbf{x})=f(\mathbf{x}')|\mathsf{Eq}]\Pr[\mathsf{Eq}]+\Pr[f(\mathbf{x})=f(\mathbf{x}')|\overline{\mathsf{Eq}}]\Pr[\overline{\mathsf{Eq}}]\\
&= \Pr[f_2(y_1,\ldots,y_{m_2})=f_2(y'_1,\ldots,y'_{m_2})|\mathsf{Eq}]\Pr[\mathsf{Eq}]\\
&\quad+\Pr[f_2(y_1,\ldots,y_{m_2})=f_2(y'_1,\ldots,y'_{m_2})|\overline{\mathsf{Eq}}]\Pr[\overline{\mathsf{Eq}}]\\
&= \Pr[\mathsf{Eq}]+\Pr[f_2(y_1,\ldots,y_{m_2})=f_2(y'_1,\ldots,y'_{m_2})|\overline{\mathsf{Eq}}]\Pr[\overline{\mathsf{Eq}}]\\
&\leq \Pr\left[\bigwedge_{i=1}^{m_2}(y_i=y'_i)\right]+\varepsilon_2\\
&\leq \min_{1\leq i\leq m_2}(\Pr[y_i=y'_i)])+\varepsilon_2\\
&\leq \varepsilon_1+\varepsilon_2.
\end{aligned}$$

In the above, we have used $\Pr[f_2(y_1,\ldots,y_m)=f_2(y'_1,\ldots,y'_m)|\mathsf{Eq}]=1$ and $\Pr[f_2(y_1,\ldots,y_m)=f_2(y'_1,\ldots,y'_m)|\overline{\mathsf{Eq}}]\Pr[\overline{\mathsf{Eq}}]\leq\Pr[f_2(y_1,\ldots,y_m)=f_2(y'_1,\ldots,y'_m)|\overline{\mathsf{Eq}}]\leq\varepsilon_2$, the last inequality following from the fact that ε_2 is a collision bound for f_2. $\square$

A generalization of this result can be obtained using essentially the same proof as given above.

Proposition 1. *Let $g_1, \ldots, g_\ell$ be random (not necessarily independent) functions with $g_i : \mathbb{F}^{m_i} \rightarrow \mathbb{F}$ having collision bound ε_i. Let g be a random function which is independent of $g_1, \ldots, g_\ell$ with $g : \mathbb{F}^\ell \rightarrow \mathbb{F}$ and having collision bound ε. Let $m = m_1 + \cdots + m_\ell$ and define a random function $h : \mathbb{F}^m \rightarrow \mathbb{F}$ by*

$$h(\mathbf{x}_1, \ldots, \mathbf{x}_\ell) = g(f_1(\mathbf{x}_1), \ldots, f_\ell(\mathbf{x}_\ell)).$$

Then a collision bound for h is $\varepsilon + \min_{1 \leq i \leq \ell} \varepsilon_i$.

Multi-Level Hashing Scheme. Let $\mathbb{F}$ be a finite field and s and t be integers such that $2^s \leq \#\mathbb{F} \leq 2^t$. Any s-bit element can be encoded into an element of $\mathbb{F}$ and any element of $\mathbb{F}$ can be encoded into a t-bit string. The values of s and t would depend on the choice of $\mathbb{F}$ and the representation of the elements of $\mathbb{F}$. If $\mathbb{F}$ is a binary extension field, then we can also have $s = t$ and $\mathbb{F} = GF(2^t)$. Given an s-bit string str, an injective function toElem encodes str into an element of $\mathbb{F}$; similarly, given an element $z \in \mathbb{F}$, an injective function toStr encodes z into a t-bit string.

Messages to be hashed are bit strings of lengths greater than or equal to zero. Given a message msg of length n bits, an injective function pad maps $\mathsf{msg} \mapsto \mathsf{msg}||0^k||\mathsf{bin}_s(n)$, where k is the minimum non-negative integer such that $n+k$ is a multiple of s and $\mathsf{bin}_s(n)$ is the s-bit binary representation of n. Suppose $\mathsf{pad}(\mathsf{msg}) = \mathsf{str}_1|| \cdots ||\mathsf{str}_r$ for some $r \geq 1$ and each str_i is a string of length s. The function $\mathsf{parse}(\mathsf{msg})$ encodes each str_i into an element of $\mathbb{F}$, i.e., $\mathsf{parse}(\mathsf{msg}) = (\mathsf{toElem}(\mathsf{str}_1), \ldots, \mathsf{toElem}(\mathsf{str}_r))$. For $\mathbf{x} \in \mathbb{F}^r$, let $\#\mathbf{x} = r$.

Let m be a fixed positive integer and let ℓ be an integer such that for any message msg, $\#\mathsf{parse}(\mathsf{msg}) \leq m^\ell$. Let $f_1, \ldots, f_\ell$ be mutually independent random functions where each $f_i : D_m \rightarrow \mathbb{F}$ with $D_m = \cup_{i=1}^m \mathbb{F}^i$ having collision bound ε_i for *equal length* messages. Here ε_i could depend on m and for polynomial hashing it indeed does. Note that $1/2^t \leq 1/\#\mathbb{F} \leq \varepsilon_i$ for $1 \leq i \leq \ell$. We also need a random function $\psi : \{1, \ldots, \ell\} \rightarrow GF(2^t)$ such that for $1 \leq i, j \leq \ell$ with $i \neq j$, $\Pr[\psi(i) \oplus \psi(j) = \alpha] = 1/2^t$ for any $\alpha \in GF(2^t)$. This ψ is to be independent of $f_1, \ldots, f_\ell$. The functions $f_1, \ldots, f_\ell$ and ψ can be instantiated as mentioned in Section 2.

Given a function $f : D_m \rightarrow \mathbb{F}$, the function $\mathsf{reduce}(f, \mathbf{x})$ with $\mathbf{x} \in \mathbb{F}^i$ for some i, is defined in Figure 1. Using reduce and a sequence of functions $\mathbf{f} = (f_1, \ldots, f_\ell)$ as mentioned above, we define a hash function $\mathsf{hash}((\mathbf{f}, \psi), \mathsf{msg})$ as given in Figure 2. The message msg to be hashed is a bit string of length greater than or equal to zero.

Note that reduce does not actually need to count the number of blocks in $\mathbf{x}$; the requirement is to divide $\mathbf{x}$ into m-element groups with the last group possibly having less than m elements. Online processing is possible using the following strategy. Start the computation at the lowest level. After m elements have been processed, one element of the next level is available for processing. Each group of m elements at the bottom level gives rise to one element at the next level and hence, the processing at the next level can proceed in a synchronised manner with that of the bottom level. The rate at which this level processes will be $1/m$ times the rate at which the bottom level processes. This is not particular to the bottom and the last-but-one level. The same strategy can be applied to higher levels, i.e., the processing at level i proceeds at a rate of $1/m$ of the processing at level $i - 1$. The requirement will be at most ℓ variables to store the intermediate values at the ℓ levels and additionally ℓ counters, one for each level, to keep track of the fact that a group of m elements have been processed at that level. The counter for each level would have to be reset to 0 after every group of m elements have been processed at that level. In the case where each f_i is instantiated using poly, the algorithm for online computation is shown in Figure 3. The memory requirement for processing a message having $r = \#\mathsf{parse}(\mathsf{msg})$ is proportional to $2\lceil \log_m r \rceil$.

Fig. 1. Definition of the function $\mathsf{reduce}(f, \mathbf{x})$.

```
$\mathsf{reduce}(f, \mathbf{x})$:
1. parse $\mathbf{x}$ as $\mathbf{x} = (\mathbf{x}_1, \ldots, \mathbf{x}_{k-1}, \mathbf{x}_k)$,
   where $\#\mathbf{x}_i = m$ for $1 \le i \le k-1$ and $1 \le \#\mathbf{x}_k \le m$;
2. return $(f(\mathbf{x}_1), \ldots, f(\mathbf{x}_{k-1}), \ldots, f(\mathbf{x}_k))$.
```

Fig. 2. Definition of multi-level hashing.

```
$\mathsf{hash}((\mathbf{f}, \psi), \mathsf{msg})$:
1. let $\mathsf{parse}(\mathsf{msg})$ equal $\mathbf{x} \in \mathbb{F}^r$;
2. $i = 1$; $\mathbf{z} = \mathbf{x}$;
3. while $\#\mathbf{z} > m$ do
4.     $\mathbf{z} = \mathsf{reduce}(f_i, \mathbf{z})$; $i = i + 1$;
5. end do;
6. return $\mathsf{toStr}(f_i(\mathbf{z})) \oplus \psi(r)$.
```

Fig. 3. On-line computation of $\mathsf{hash}((\mathbf{f} = (f_1, \ldots, f_\ell), \psi), \mathsf{msg})$ where f_i is poly_{α_i} and the key to ψ is α. Variables required: $\mathsf{cnt}_1, \ldots, \mathsf{cnt}_{\ell-1}$; $R_1, \ldots, R_\ell$. The variable j records the maximum value of the level currently reached in the algorithm. The sub-routine $\mathsf{nextBlk}$ will read the message and return the next m elements of $\mathsf{parse}(\mathsf{msg})$, or possibly less than m elements for the last block.

```
1.  $\mathsf{cnt}_1 = \cdots = \mathsf{cnt}_\ell = 0$;
2.  $R_1 = \cdots = R_\ell = 0$;
3.  $\beta = \alpha$; $j = 1$;
4.  while $\mathsf{not}(\mathsf{endof}(\mathsf{msg}))$ do
5.      $\mathbf{u} = \mathsf{nextBlk}()$;
6.      for $i = 1$ to $\#\mathbf{u}$ do $\beta = \psi(\beta)$;
7.      $v = \mathsf{poly}_{\alpha_1}(\mathbf{u})$; $R_1 = \alpha_2 R_1 + v$;
8.      $\mathsf{cnt}_1 = \mathsf{cnt}_1 + 1$; $\mathsf{flg} = \mathsf{true}$; $i = 1$;
9.      while $\mathsf{flg}$ is $\mathsf{true}$ do
10.         if $(\mathsf{cnt}_i = m)$
11.             if $(i > j)$ then $j = i$;
12.             $v = R_i$; $R_i = 0$; $\mathsf{cnt}_i = 0$;
13.             $R_{i+1} = \alpha_{i+2} R_{i+1} + v$;
14.             $\mathsf{cnt}_{i+1} = \mathsf{cnt}_{i+1} + 1$;
15.             $i = i + 1$;
16.         else $\mathsf{flg} = \mathsf{false}$;
17.     end do;
18. end do;
19. return $\mathsf{toStr}(R_j) \oplus \beta$.
```

Theorem 1. *Let* $\mathsf{msg}, \mathsf{msg}'$ *be messages of lengths* $n, n' \geq 0$ *such that*

1. $\mathsf{msg} \neq \mathsf{msg}'$,
2. $1 \leq \#\mathsf{parse}(\mathsf{msg}), \#\mathsf{parse}(\mathsf{msg}') \leq m^\ell$, *and*
3. $y = \mathsf{hash}((\mathbf{f}, \psi), \mathsf{msg})$, $y' = \mathsf{hash}((\mathbf{f}, \psi), \mathsf{msg})$.

Then $\Pr[y = y'] \leq \varepsilon_1 + \cdots + \varepsilon_\ell$ *where* ε_i *is the collision probability of* f_i.

Note. The requirement on f_i is that they are AU for equal length messages, whereas the theorem states that hash is AU even for unequal length messages.

Proof: Let $\mathbf{x} = \mathsf{parse}(\mathsf{msg})$, $r = \#\mathbf{x}$ and j be such that $m^{j-1} \leq r \leq m^j$. Then the functions $f_1, \ldots, f_j$ are used by hash on input msg. In the algorithm to compute hash, let $\mathbf{z}_0 = \mathbf{x}$ and denote the output of reduce in the ith iteration by $\mathbf{z}_i$, i.e.,

$$\mathbf{x} = \mathbf{z}_0, \mathbf{z}_1 = \mathsf{reduce}(f_1, m, \mathbf{z}_0), \ldots, \mathbf{z}_j = \mathsf{reduce}(f_j, m, \mathbf{z}_{j-1}).$$

The primed variables denote the similar quantities for msg'.

Case $r \neq r'$. In this case,

$$\begin{aligned}
\Pr[y = y'] &= \Pr[\mathsf{toStr}(\mathbf{z}_j) \oplus \psi(r) = \mathsf{toStr}(\mathbf{z}'_{j'}) \oplus \psi(r')] \\
&= \Pr[\psi(r) \oplus \psi(r') = \mathsf{toStr}(\mathbf{z}_j) \oplus \mathsf{toStr}(\mathbf{z}'_{j'})] \\
&\overset{(a)}{=} \frac{1}{2^t} \\
&\leq \frac{1}{|\mathbb{F}|} \leq \epsilon_1 + \cdots + \epsilon_\ell.
\end{aligned}$$

The step (a) follows form the XOR universal property of ψ.

Case $r = r'$. This implies that $j = j'$ and from the definition of hash, $y = y'$ if and only if $\mathbf{z}_j = \mathbf{z}'_j$.

The lengths of $\mathsf{pad}(\mathsf{msg})$ and $\mathsf{pad}(\mathsf{msg}')$ are equal; if $n \neq n'$, then the last s bits of $\mathsf{pad}(\mathsf{msg})$ and $\mathsf{pad}(\mathsf{msg}')$ are not equal; if $n = n'$, then the equal length messages msg and msg' themselves are not equal. In both cases, we have $\mathbf{x} \neq \mathbf{x}'$.

For $0 \leq i \leq j$, let Eq_i be the event $\mathbf{z}_i = \mathbf{z}'_i$. We are interested in the event Eq_j and since $\mathbf{z}_0 = \mathbf{x} \neq \mathbf{x}' = \mathbf{z}'_0$, the probability of Eq_0 is 0. Write $\mathbf{z}_i = (\mathbf{z}_{i,1}, \ldots, \mathbf{z}_{i,l_i})$ with $\#\mathbf{z}_{i,k} = m$ for $1 \leq k \leq l_i - 1$ and $1 \leq \#\mathbf{z}_{i,l_i} \leq m$. Then we can write $\mathbf{z}_{i+1} = (y_{i+1,1}, \ldots, y_{i+1,l_i})$ where $y_{i+1,k} = f_{i+1}(\mathbf{z}_{i,k})$. Since $r = r'$, we have $l_i = l'_i$ for $1 \leq i \leq \ell$. Now the event Eq_{i+1} can be written as $\bigwedge_{k=1}^{l_i} \left(y_{i+1,k} = y'_{i+1,k}\right)$. The event $\overline{\mathsf{Eq}_i}$ means that $(\mathbf{z}_{i,1}, \ldots, \mathbf{z}_{i,l_i}) \neq (\mathbf{z}'_{i,1}, \ldots, \mathbf{z}'_{i,l_i})$ so that for at least one k, $\mathbf{z}_{i,k} \neq \mathbf{z}'_{i,k}$ and then for this k, $\Pr[f_{i+1}(\mathbf{z}_{i,k}) = f_{i+1}(\mathbf{z}'_{i,k})] \leq \varepsilon_{i+1}$ by the collision bound on f_{i+1}. So,

$$\begin{aligned}
\Pr\left[\bigwedge_{k=1}^{l_i} (y_{i+1,k} = y'_{i+1,k}) | \overline{\mathsf{Eq}_i}\right] &= \Pr\left[\bigwedge_{k=1}^{l_i} (f_{i+1}(\mathbf{z}_{i,k}) = f_{i+1}(\mathbf{z}'_{i,k})) | \overline{\mathsf{Eq}_i}\right] \\
&\leq \min_{1 \leq k \leq l_i} \Pr[f_{i+1}(\mathbf{z}_{i,k}) = f_{i+1}(\mathbf{z}'_{i,k}) | \overline{\mathsf{Eq}_i}] \\
&\leq \varepsilon_{i+1}. \qquad (2)
\end{aligned}$$

For $0 \leq i \leq j-1$,

$$\begin{aligned}
\Pr[\mathsf{Eq}_{i+1}] = \Pr[\mathbf{z}_{i+1} = \mathbf{z}'_{i+1}] &= \Pr\left[\bigwedge_{k=1}^{l_i} (y_{i+1,k} = y'_{i+1,k})\right] \\
&= \Pr\left[\bigwedge_{k=1}^{l_i} (y_{i+1,k} = y'_{i+1,k}) | \mathsf{Eq}_i\right] \Pr[\mathsf{Eq}_i] \\
&\quad + \Pr\left[\bigwedge_{k=1}^{l_i} (y_{i+1,k} = y'_{i+1,k}) | \overline{\mathsf{Eq}_i}\right] \Pr[\overline{\mathsf{Eq}_i}] \\
&= \Pr[\mathsf{Eq}_i] + \Pr\left[\bigwedge_{k=1}^{l_i} (y_{i+1,k} = y'_{i+1,k}) | \overline{\mathsf{Eq}_i}\right] \Pr[\overline{\mathsf{Eq}_i}] \\
&\leq \Pr[\mathsf{Eq}_i] + \Pr\left[\bigwedge_{k=1}^{l_i} (y_{i+1,k} = y'_{i+1,k}) | \overline{\mathsf{Eq}_i}\right] \\
&\leq \Pr[\mathsf{Eq}_i] + \varepsilon_{i+1}.
\end{aligned}$$

The last step follows from (2). Note that the above calculation is similar to the one given in the proof of Lemma 1. Extending this we obtain

$$\Pr[\mathsf{Eq}_j] \leq \Pr[\mathsf{Eq}_{j-1}] + \varepsilon_j \leq \cdots \leq \Pr[\mathsf{Eq}_0] + \varepsilon_j + \cdots + \varepsilon_1 \leq \varepsilon_j + \cdots + \varepsilon_1.$$

□

XOR Universal. The hash function defined in Figure 2 provides almost universality. For certain applications, the requirement is to obtain XOR universality. It is possible to modify the algorithm to obtain XOR universality. Suppose each f is instantiated using either poly or BRW. As mentioned earlier, both $\alpha\mathsf{poly}_\alpha$ and $\alpha\mathsf{BRW}_\alpha$ are XOR universal. We refer to these variants by f^{XOR}. Suppose in algorithm hash, $m^{j-1} < r \leq m^j$. To obtain XOR universality, in Figure 2, the last line is changed to

$$\text{return } \mathsf{toStr}(f_i^{\mathrm{XOR}}(\mathbf{z})) \oplus \psi(r).$$

This means that the message is processed with $f_1, \ldots, f_{j-1}, f_j^{\mathrm{XOR}}$ instead of being processed with $f_1, \ldots, f_{j-1}, f_j$. In other words, for the first $(j-1)$ layers the usual functions are used, while for the last layer we use the variant which is XOR universal. It is not difficult to show that this achieves XOR universality of the entire construction.

Instantiating ψ and $\mathbf{f} = (f_1, \ldots, f_\ell)$. Let $\alpha_1, \ldots, \alpha_\ell$ be independent and uniform random elements of $\mathbb{F}$. Further, let α be a uniform random element of $GF(2^t)$ which is independent of $\alpha_1, \ldots, \alpha_\ell$.

The key for the function ψ is α and f_i is instantiated as either poly_{α_i} or as $\mathsf{BRW}\alpha_i$. The collision bound and the number of multiplications required depend on the instantiations of f_i.

Case f as poly. In this case, each f_i has collision bound $(m-1)/|\mathbb{F}|$ and from Theorem 1, the collision bound for $\mathsf{hash}((\mathbf{f}, \psi), \mathsf{msg})$ is $\ell(m-1)/|\mathbb{F}|$.

Proposition 2. *Suppose that in Figure 2, f_i is instantiated as poly_{α_i}. Then the number of multiplications required by* hash *to process* msg *with* $\#\mathsf{parse}(\mathsf{msg}) = r$ *equals* $r - 1$.

Proof: Let $n_0 = \mathsf{blks}(\mathbf{x})$. The function hash defines the sequence of integers $n_0, n_1, \ldots, n_s, n_{s+1}$ where $n_i = (n_{i+1} - 1)m + r_i$, $0 \leq i \leq r - 1$ with $n_s = 1$ and $1 \leq r_1, \ldots, r_s \leq m$. At the ith step, the number of multiplications required is $a_i = (n_i - 1)(m - 1) + r_i - 1 = (n_{i-1} - 1) - (n_i - 1)$. The total number of multiplications is equal to $a_1 + a_2 + \cdots + a_s = n_0 - 1$. $\square$

Case f as BRW. Using Theorem 1, the collision bound is $\ell(2m - 1)/|\mathbb{F}|$ and the number of multiplications required to process msg with $\mathsf{parse}(\mathsf{msg}) = r$ is at most equal to

$$\frac{n}{2} \times \left(\frac{m^{j+1} - 1}{m^j(m - 1)}\right) = \frac{n}{2} \times \left(1 + \frac{1}{m - 1}\left(\frac{m^j - 1}{m^j}\right)\right).$$

Here j is the unique positive integer such that $m^{j-1} < r \leq m^j$.

Let $n_0 = n$ and $n_i = \lceil n_{i-1}/m \rceil$ for $0 < i \leq s$ with $n_s = 1$. The number of multiplications equals

$$\left\lfloor \frac{m}{2} \right\rfloor \sum_{i=1}^{j} (n_i - 1) + \left\lfloor \frac{r_1}{2} \right\rfloor + \cdots + \left\lfloor \frac{r_s}{2} \right\rfloor \leq \left\lfloor \frac{n_0}{2} \right\rfloor + \left\lfloor \frac{n_1}{2} \right\rfloor + \cdots + \left\lfloor \frac{n_{s-1}}{2} \right\rfloor.$$

If $n_0 = m^j$ for some j, then the number of multiplications required is $\frac{m}{2}\left(\frac{m^j - 1}{m - 1}\right)$.

Combination of poly and BRW. It is possible to combine the two options, i.e., instantiate some of the f's using poly and the other f's using BRW. One reason for doing this could be the issue of pre-computation. The computation of poly_α can be speeded up using a pre-computed table based on α and this cannot be done for BRW_α. So, one option is to use poly to instantiate f_1 while $f_2, \ldots, f_\ell$ are instantiated using BRW. Since most of the multiplications are done at the lowest level, the pre-computed table can be used to speed-up these multiplications. Further, keeping multiplication tables for each α_i can be costly in terms of storage and so one can use BRW hashing for the higher levels since this requires lesser number of multiplications.

If each f_i is instantiated with either poly or BRW, then the final value at the end of the while loop in Figure 2 can be written as a polynomial in the variables $\alpha_1, \ldots, \alpha_\ell$. Viewed differently, the construction can be considered to be using multi-variate polynomials to reduce the collision bound.

4 Conclusion

A new universal hash construction has been described. For hashing messages of a maximum specified length and achieving a desired security level, the new algorithm improves over usual polynomial hashing by reducing the size of the underlying field over which computations take place. This can lead to efficiency improvement in the speed of hashing. The trade-off is an increase in the key size.

References

1. Daniel J. Bernstein. The Poly1305-AES message-authentication code. In Henri Gilbert and Helena Handschuh, editors, *FSE*, volume 3557 of *Lecture Notes in Computer Science*, pages 32–49. Springer, 2005.
2. Daniel J. Bernstein. Polynomial evaluation and message authentication, 2007. `http://cr.yp.to/papers.html#pema`.

3. Jürgen Bierbrauer, Thomas Johansson, Gregory Kabatianskii, and Ben J. M. Smeets. On families of hash functions via geometric codes and concatenation. In Douglas R. Stinson, editor, *CRYPTO*, volume 773 of *Lecture Notes in Computer Science*, pages 331–342. Springer, 1993.
4. John Black, Shai Halevi, Hugo Krawczyk, Ted Krovetz, and Phillip Rogaway. UMAC: Fast and secure message authentication. In Michael J. Wiener, editor, *CRYPTO*, volume 1666 of *Lecture Notes in Computer Science*, pages 216–233. Springer, 1999.
5. Larry Carter and Mark N. Wegman. Universal classes of hash functions. *J. Comput. Syst. Sci.*, 18(2):143–154, 1979.
6. Edgar N. Gilbert, F. Jessie MacWilliams, and Neil J. A. Sloane. Codes which detect deception. *Bell System Technical Journal*, 53:405–424, 1974.
7. Shai Halevi and Hugo Krawczyk. MMH: Software message authentication in the gbit/second rates. In Eli Biham, editor, *Fast Software Encryption*, volume 1267 of *Lecture Notes in Computer Science*, pages 172–189. Springer, 1997.
8. Michael O. Rabin and Shmuel Winograd. Fast evaluation of polynomials by rational preparation. *Communications on Pure and Applied Mathematics*, 25:433–458, 1972.
9. Phillip Rogaway. Bucket hashing and its application to fast message authentication. *J. Cryptology*, 12(2):91–115, 1999.
10. Palash Sarkar. A general mixing strategy for the ECB-Mix-ECB mode of operation. *Information Processing Letters.* To appear.
11. Victor Shoup. On fast and provably secure message authentication based on universal hashing. In Neal Koblitz, editor, *CRYPTO*, volume 1109 of *Lecture Notes in Computer Science*, pages 313–328. Springer, 1996.
12. Douglas R. Stinson. Universal hashing and authentication codes. *Des. Codes Cryptography*, 4(4):369–380, 1994.
13. Mark N. Wegman and Larry Carter. New hash functions and their use in authentication and set equality. *J. Comput. Syst. Sci.*, 22(3):265–279, 1981.