Polynomial evaluation
and message authentication

D. J. Bernstein
University of Illinois at Chicago
Cost of this algorithm:
5 mults, 4 adds.

Output of this algorithm,
given $m_1, \ldots, r_1, \ldots \in \mathbb{F}_q$:
$m_1 r_1 + \cdots + m_5 r_5$. 
Alternative (1968 Winograd),
\[ \approx 2 \times \text{speedup in matrix mult:} \]
\[
\begin{array}{cccccccccccc}
m_1 & r_2 & m_2 & r_1 & m_3 & r_4 & m_4 & r_3 & m_5 & r_5 \\
\downarrow & \downarrow & \downarrow & \downarrow & \downarrow & \downarrow & \downarrow & \downarrow & \downarrow & \downarrow \\
+ & + & + & + & + & \times \\
\end{array}
\]

Output in \( F_q[m_1, \ldots, r_1, \ldots] \):
\[
m_5 r_5 + (m_3 + r_4)(m_4 + r_3) + (m_1 + r_2)(m_2 + r_1) = m_1 r_1 + m_2 r_2 + m_3 r_3 + m_4 r_4 + m_5 r_5 + m_1 m_2 + m_3 m_4 + r_1 r_2 + r_3 r_4.
\]
One good way to recognize forged/corrupted messages:

Standardize a prime $p = 1000003$.

Sender rolls 10-sided die to generate independent uniform random secrets
$r_1 \in \{0, 1, \ldots, 999999\}$,
$r_2 \in \{0, 1, \ldots, 999999\}$,
\ldots,
$r_5 \in \{0, 1, \ldots, 999999\}$,
$s_1 \in \{0, 1, \ldots, 999999\}$,
\ldots,
$s_{100} \in \{0, 1, \ldots, 999999\}$.
Sender meets receiver in private and tells receiver the same secrets $r_1, r_2, \ldots, r_5, s_1, \ldots, s_{100}$.

Later: Sender wants to send 100 messages $m_1, \ldots, m_{100}$, each $m_n$ having 5 components $m_{n,1}, m_{n,2}, m_{n,3}, m_{n,4}, m_{n,5}$ with $m_{n,i} \in \{0, 1, \ldots, 999999\}$.

Sender transmits 30-digit $m_{n,1}, m_{n,2}, m_{n,3}, m_{n,4}, m_{n,5}$ together with an authenticator $(m_{n,1}r_1 + \cdots + m_{n,5}r_5 \mod p) + s_n \mod 1000000$ and the message number $n$. 
e.g. \( r_1 = 314159, r_2 = 265358, \)
\( r_3 = 979323, r_4 = 846264, \)
\( r_5 = 338327, s_{10} = 950288, \)
\( m_{10} = 000006 000007 000000 000000 000000: \)

Sender computes authenticator
\[
(6r_1 + 7r_2 \mod p) \\
+ s_{10} \mod 1000000 = \\
(6 \cdot 314159 + 7 \cdot 265358 \\
\mod 1000003) \\
+ 950288 \mod 1000000 = \\
742451 + 950288 \mod 1000000 = \\
692739.
\]

Sender transmits
\[10 000006 000007 000000 000000 000000 692739.\]
Main work is multiplication. For each 6-digit message chunk, have to do one multiplication by a 6-digit secret $r_i$.

Scaled up for serious security: Choose, e.g., $p = 2^{130} - 5$. For each 128-bit message chunk, have to do one multiplication by a 128-bit secret $r_i$. Reduce output mod $2^{130} - 5$. $\approx 5$ cycles per message byte, depending on CPU.

Many papers on choosing fields, computing products quickly.
Provably secure authenticators
\((m_1 r_1 + m_2 r_2 + \cdots) + s\): 1974 Gilbert/MacWilliams/Sloane.

1999 Black/Halevi/Krawczyk/Krovetz/Rogaway (crediting unpublished Carter/Wegman, failing to credit Winograd):
Replace \(m_1 r_1 + m_2 r_2\)
with \((m_1 + r_1)(m_2 + r_2)\),
replace \(m_3 r_3 + m_4 r_4\)
with \((m_3 + r_3)(m_4 + r_4)\), etc.
Half as many multiplications for each message chunk.
Expand short key $k$ into long secret $r_1, \ldots, s_1, \ldots$
as, e.g., $AES_k(1), AES_k(2), \ldots$

Oops, not uniform random. But easily prove that attack implies attack on AES.

Generate $r$’s, $s$’s on demand? Need $\ell + 1$ AES invocations for $r_1, r_2, \ldots, r_\ell, s_n$.

Cache $r_1, r_2, \ldots, r_\ell$? Bad performance for large $\ell$:
huge initialization cost;
many expensive cache misses;
too big for low-cost hardware.
1979 Wegman/Carter: Another authentication function, fewer secrets $r_1, r_2, \ldots$.

1987 Karp/Rabin, 1981 Rabin: Another authentication function, extremely short secret $r$, but expensive to generate.

Horner’s rule (const coeff 0):

\[ r \rightarrow m_5 \rightarrow m_4 \rightarrow m_3 \rightarrow m_2 \rightarrow m_1 \]
Cost of this algorithm:
5 mults, 4 adds,
just like dot product.

Output in
\[ F_q[m_1, m_2, m_3, m_4, m_5, r]: m_5 r^5 + m_4 r^4 + \cdots + m_1 r. \]

Substituting any message
\((m_1, m_2, m_3, m_4, m_5) \in F_q^5\)
produces poly in \(F_q[r];\)
message \(\mapsto\) poly is injective.

Secure for authentication:
at most 5 values of \(r\) are roots
of any shifted difference
of polys for distinct messages.
1 multiplication per chunk. Can we do better?

Classic observation (1955 Motzkin, 1958 Belaga, et al.): For each $\varphi \in \mathbb{C}[r]$ there is an algorithm that computes $\varphi$ using $\approx (\deg \varphi)/2$ multiplications.

Idea: 
$$((ar + b)(r^2 + c) + d)$$
$$r^2 + e + f(r^2 + g) + h.$$ 

Doesn’t solve the authentication problem. This set of algorithms maps surjectively but not injectively to $\mathbb{C}[r]$. 
1970 Winograd: Can achieve 
$\approx (\text{deg } \varphi)/2$ multiplications 
with “rational preparation,”
i.e., rational map $\varphi \mapsto$ algorithm.

Idea: 
\[(r + a)(r^2 + b) + r + c)\]
\[(r^4 + d) + (r + e)(r^2 + f) + r + g.\]
Adapt idea to non-monic $\varphi$
and to $\text{deg } \varphi \notin \{1, 3, 7, 15, \ldots\}$. 

“Aha! 
\[(r + a)(r^2 + b) + r + c)\]
\[(r^4 + d) + (r + e)(r^2 + f) + r + g\]
is an authenticator of 
message $(a, b, c, d, e, f, g)$."

Have to be careful. Injective? 
Not just for fixed degree?
Fix odd prime $p$. Define $H : \{0, 2, 4, \ldots, p - 3\}^* \to \mathbb{F}_p[r]$ by $H() = 0$; $H(m_1) = r + m_1$;
$H(m_1, \ldots, m_\ell) =$
$H(m_{t+1}, \ldots, m_\ell) +$
$(r^t + m_t)H(m_1, \ldots, m_{t-1})$ if $t \in \{2, 4, 8, 16, \ldots\}$, $t \leq \ell < 2t$.

E.g. $H(m_1, m_2) =$
$(r + m_1)(r^2 + m_2);$  
$H(m_1, m_2, m_3) =$
$(r + m_1)(r^2 + m_2) + (r + m_3).$

(Could change $H()$ to 1,
avoid special case for $\ell = 1$.  
But my $H$ is slightly faster.)
Easy to prove: $H$ is injective.

Use $rH(m) + s_n$ as authenticator of $n$th message $m$.

(Good choice of $p$: $2^{107} - 1$. Put 13 bytes into each chunk.)

Combines all the advantages of previous authenticators: extremely short secret $r$, trivial to generate; 1/2 multiplications per chunk.