Leftovers from Lecture 3
Multiplication: Polynomial multiplication, and then remainder modulo the defining polynomial \( f(x) \):

\[
(1,1,0,1,1) \ast (0,1,0,1,1) = (1,1,0,0,1)
\]

For small size finite field, a lookup table is the most efficient method for implementing multiplication.
Implementing $GF(2^5)$ in XMAPLE

Irreducible polynomial

```maple
> G32 := GF(2, 5, x^5 + x^4 + x^3 + x + 1):
> a := G32[ConvertIn](x);
    a := x

> b := G32[`^`](a, 8): # colon at end of statement supresses printing
> c := G32[`^`](a, 9):  # canonical representation, higher momonials to the left
> G32[ConvertOut](b);
> G32[ConvertOut](c);

x^3 + x^2 + x + 1
x^4 + x^3 + x^2 + x
```
More \textit{GF}(2^5) \textit{Operations in XMAPLE}

\begin{verbatim}
> d := G32[\textbackslash`+\textbackslash`](b,c):
   G32[ConvertOut](d);
   x^4 + 1
> G32[isPrimitiveElement](d);
   true
> e:=G32[\textbackslash`\wedge\textbackslash`](a,-1):
   G32[ConvertOut](e);
   x^4 + x^3 + x^2 + 1
> G32[\textbackslash`\wedge\textbackslash`](a,e);
   1

> for i from 1 to 32 do
   f:= G32[\textbackslash`\wedge\textbackslash`](a,i):
   print(f, G32[isPrimitiveElement](f))
end do:
   x, true
   x^2, true
   x^3, true
   x^4, true
   1 + x + x^3 + x^4, true
   1 + x^2 + x^3, true
   x + x^3 + x^4, true
\end{verbatim}

Addition: \(b+c\)

test primitive element \(e \leftarrow \text{inverse of } a\)

Multiplication: \(a \cdot e\)

Loop for finding primitive elements
LECTURE 4

Data Integrity & Authentication

Message Authentication Codes (MACs)
Goal

Ensure integrity of messages, even in presence of an active adversary who sends own messages.

Remark: Authentication is orthogonal to secrecy, yet systems often required to provide both.
Definitions

• Authentication algorithm - A
• Verification algorithm - V ("accept"/"reject")
• Authentication key - k
• Message space (usually binary strings)
• Every message between Alice and Bob is a pair (m, A_k(m))
• A_k(m) is called the authentication tag of m
Definition (cont.)

• Requirement – $V_k(m, A_k(m)) = \text{“accept”}$
  - The authentication algorithm is called MAC (Message Authentication Code)
  - $A_k(m)$ is frequently denoted $MAC_k(m)$
  - Verification is by executing authentication on $m$ and comparing with $MAC_k(m)$
Properties of MAC Functions

• Security requirement - adversary can’t construct a new legal pair \((m, MAC_k(m))\) even after seeing \((m_i, MAC_k(m_i))\) \((i=1,2,...,n)\)

• Output should be as short as possible

• The MAC function is not 1-to-1
Adversarial Model

• Available Data:
  – The MAC algorithm
  – Known plaintext
  – Chosen plaintext
• Note: chosen MAC is unrealistic
• Goal: Given n legal pairs
  \((m_1, MAC_k(m_1)), \ldots, (m_n, MAC_k(m_n))\)
  find a new legal pair \((m, MAC_k(m))\)
We will say that the adversary succeeded even if the message Fran forged is “meaningless”. The reason is that it is hard to predict what has and what does not have a meaning in an unknown context, and how will Bob, the receiver, react to such successful forgery.
Efficiency

- Adversary goal: given $n$ legal pairs $(m_1, \text{MAC}_k(m_1)), \ldots, (m_n, \text{MAC}_k(m_n))$ find a new legal pair $(m, \text{MAC}_k(m))$ efficiently and with non-negligible probability.

- If $n$ is large enough then $n$ pairs $(m_i, \text{MAC}_k(m_i))$ determine the key $k$ uniquely (with high prob.). Thus a non-deterministic machine can guess $k$ and verify it. But doing this deterministically should be computationally hard.
MACs Used in Practice

We describe a MAC based on CBC Mode Encryption, and a MAC based on cryptographic hash functions.
Reminder: CBC Mode Encryption
(Cipher Block Chaining)

Previous ciphertext is XORed with current plaintext before encrypting current block.
An initialization vector $S_0$ is used as a “seed” for the process.
Seed can be “openly” transmitted.
CBC Mode MACs

- Start with the all zero seed.
- Given a message consisting of \( n \) blocks \( M_1, M_2, \ldots, M_n \), apply CBC (using the secret key \( k \)).

\[
\begin{align*}
0000000 & \quad M_1 & \quad M_2 & \quad M_n \\
\downarrow & \quad \downarrow & \quad \downarrow & \quad \downarrow \\
E_k & \quad E_k & \quad E_k & \quad E_k \\
C_1 & \quad C_2 & \quad \ldots & \quad C_n
\end{align*}
\]

- Produce \( n \) “ciphertext” blocks \( C_1, C_2, \ldots, C_n \), discard first \( n-1 \).
- Send \( M_1, M_2, \ldots, M_n \) & the authentication tag \( MAC_k(M) = C_n \).
Security of CBC MAC [BKR]

• Claim: If $E_k$ is a pseudo random function, then CBC MAC is resilient to forgery.

• Proof outline: Assume CBC MAC can be forged efficiently. Transform the forging algorithm into an algorithm distinguishing $E_k$ from random function efficiently.
Combined Secrecy & MAC

• Given a message consisting of $n$ blocks $M_1, M_2, \ldots, M_n$, apply CBC (using the secret key $k_1$) to produce $MAC_{k_1}(M)$.

• Produce $n$ ciphertext blocks $C_1, C_2, \ldots, C_n$ under a different key, $k_2$.

• Send $C_1, C_2, \ldots, C_n$ & the authentication tag $MAC_{k_1}(M)$. 
Hash Functions

• Map large domains to smaller ranges
• Example $h: \{0, 1, \ldots, p^2\} \rightarrow \{0, 1, \ldots, p-1\}$
defined by $h(x) = ax + b \mod p$
• Used extensively for searching (hash tables)
• Collisions are resolved by several possible means – chaining, double hashing, etc.
Collision Resistance

• A hash function \( h: D \rightarrow R \) is called weakly collision resistant for \( x \in D \) if it is hard to find \( x' \neq x \) such that \( h(x') = h(x) \)

• A function \( h: D \rightarrow R \) is called strongly collision resistant if it is hard to find \( x, x' \) such that \( x' \neq x \) but \( h(x) = h(x') \)
The Birthday Paradox

• If 23 people are chosen at random the probability that two of them have the same birth-day is greater than 0.5

• More generally, let \( h: D \rightarrow R \) be any mapping. If we chose \( 1.17|R|^{1/2} \) elements of \( D \) at random, the probability that two of them are mapped to the same image is greater than 0.5.
Cryptographic Hash Functions

Cryptographic hash functions are hash functions that are strongly collision resistant.

• Notice: No secret key.

• Should be very fast to compute, yet hard to find coliding pairs (impossible if $P=NP$).

• Usually defined by:
  – Compression function mapping $n$ bits (e.g. 512) to $m$ bits (e.g. 160), $m < n$. 
Extending to Longer Strings

$H : D \rightarrow R \quad \text{(fixed sets, typically } \{0,1\}^n \text{ and } \{0,1\}^m \text{)}$
Extending the Domain (cont.)

- The seed is usually constant.
- Typically, padding (including text length of original message) is used to ensure a multiple of $n$.
- Claim: if the basic function $H$ is collision resistant, then so is its extension.
Lengths

• Input message length should be arbitrary. In practice it is usually up to $2^{64}$, which is good enough for all practical purposes.
• Block length is usually 512 bits.
• Output length should be at least 160 bits to prevent birthday attacks.
Real-World Hash Functions

• MD family (“message digest”)
  – MD-2
  – MD-4 (full description in Stinson’s book)
  – MD-5
• SHA and SHA-1 (secure hash standard, 160 bits)
  (www.itl.nist.gov/fipspubs/fip180-1.htm)
• RIPE-MD
• SHA-256, 384 and 512 (proposed standards, longer digests)
Basing MACs on Hash Functions

• First goal: combine message and secret key, hash and produce MAC
• Second goal: work with any cryptographic hash function

• First attempt: \( \text{MAC}_k(m) = h(k, m) \)
• Second attempt: \( \text{MAC}_k(m) = h(m, k) \)
HMAC

- Proposed in 1996 by [BCK]
- Receives as input a message $m$, a key $k$ and a hash function $h$
- Outputs a MAC by:
  $\text{HMAC}_k(m, h) = h(k \oplus \text{opad}, h(k \oplus \text{ipad}, m))$
- Theorem [BCK]: HMAC can be forged if and only if the underlying hash function is broken (collisions found).
HMAC in Practice

• SSL / TLS
• WTLS
• IPSec:
  – AH
  – ESP
Back to Number Theory
Quadratic Residues

- An element x is a *quadratic residue* modulo n if there exists y such that $y^2 \equiv x \mod n$
- If x is a quadratic residue then so is $-x \mod n$
- If p is prime there are exactly $(p-1)/2$ quadratic residues
- If p is prime, and g is a generator of the multiplicative group, the quadratic residues are even powers of g.
One-Way Functions

• A function \( f: D \rightarrow R \) is called one-way if:
  – Computing \( f(x) \) is “easy”
  – Computing \( f^{-1}(y) \) for almost all the images is “hard”

• Given the “real-world” definition of “hard” a one-way function may be a single function (e.g. SHA-1)

• Given the theoretical definition, we refer to a family of one-way functions
Example

- The Domain is all the pairs of prime numbers.
- The function is \( f(p, q) = pq \)
- Multiplication is easy – naïve algorithm is \( O(n^2) \)
- Factoring is difficult – simple algorithm is \( O(2^{n/2}) \). NFS and ECM are better but not polynomial.
- The function \( f(p, q) = pq \) maintains length