Hashes and Message Digests

- What hashes can do
- MD2
- MD4, MD5, SHA-1
- HMAC
Hash Functions

- Aka: message digests, one-way transformations
- Take a message $m$ of arbitrary length (transformed into a string of bits) and computes from it a fixed-length (short) number $h(m)$

Properties:
- easy-to-compute: for any message $m$, it is relatively easy to compute $h(m)$
- non-reversible: given $h(m)$, there is no way to find an $m$ that hashes to $h(m)$ except trying all possibilities of $m$
- computationally infeasible to find $m$ and $m'$ such that $h(m)=h(m')$ and $m!=m'$
Notion of Randomness

- Any particular bit in the outputs should have 50% chance to be on
- Each output should have about half the bits on, w.h.p
- Any two outputs should be completely uncorrelated, no matter how similar the inputs are
General Structure

- Repeated use of a compression function, $f$, that takes two inputs (the chaining variable and input block), and produces an $n$-bit output.
Birthday Paradox

• If there are 23 or more people in a room, there is better than 50% chance that two of them will have the same birthday

• Assume n inputs (number of people) and k possible outputs (365 days)

• If n > k^{1/2}, there is a good chance of finding a matching pair
Length of Digest

If the length of the digest is n-bit long:

- It takes $O(2^n)$ to find a message with a given digest
- It takes $O(2^{n/2})$ to find two messages with the same digest (the length of digest should be sufficient to resist this attack)
Length of Digest (Cont’d)

• Due to birthday attack, the length of the digest in general should be twice the length of the key in block ciphers
• For example, SHA-256, SHA-384, and SHA-512 to match the key lengths 128, 192, and 256 in AES
What Hashes Can Do

- Authentication
- Integrity check (MAC)
- Encryption
Authentication

Authentication with SKC (previous example)

Authentication with message digest
Message Authentication Code (MAC)

- MD ($K_{AB}||m$): only those knowing the secret $K_{AB}$ can compute/verify the MAC *(Problem?)*
- See “Problem With Keyed Hash” and “Solutions”
Encryption

- Generate one-time pad: (similar to OFB)
  \[b_1 = \text{MD}(K_{AB}|IV), \quad b_2 = \text{MD}(K_{AB}|b_1), \quad b_3 = \text{MD}(K_{AB}|b_2), \ldots\]
- XOR the message with the one-time pad bit sequence (Problem with one-time pad?)
- Mixing in the plaintext: (similar to CFB)
  \[c_1 = m_1 \text{ XOR } b_1, \quad c_2 = m_2 \text{ XOR } b_2, \quad c_3 = m_3 \text{ XOR } b_3, \ldots\]
Replacing Hash with SKC

- Unix password hash: password is converted into a secret key, which is input into DES to encrypt the constant 0, producing a hashed password
- Hashing large messages: (Problem? See Figure 5-2, pp. 128)
MD2, MD4, MD5, SHA-1

Message digest (MD), designed by Ron Rivest
• MD2: 1989, operates on 8-bit octets
• MD4: 1990, operates on 32-bit words
• MD5: 1991, operates on 32-bit words

Secure hash algorithm (SHA), proposed by NIST
• SHA-1: 1995, operates on 32-bit words
MD2

- Input: arbitrary number of octets
- Output: 128-bit message digest
- Step 1: Pad the message to be a multiple of 16 octets
- Step 2: Append a 16-octet checksum to the message
- Step 3: Process the message, 16 octets at a time, to produce the message digest
Step 1: Padding

• 16 octets of padding are added if the message is initially a multiple of 16 octets
• Otherwise, \( r \) octets of padding are added to make the message a multiple of 16 octets
Step 2: Checksum Computation

- Checksum: 16-octet quantity, appended to the message $m$
- $m|\text{checksum}$ is processed by MD2 to obtain the actual message digest
### π Substitution Table

<table>
<thead>
<tr>
<th>41</th>
<th>46</th>
<th>67</th>
<th>201</th>
<th>162</th>
<th>216</th>
<th>124</th>
<th>1</th>
<th>61</th>
<th>54</th>
<th>84</th>
<th>236</th>
<th>240</th>
<th>6</th>
<th>19</th>
</tr>
</thead>
<tbody>
<tr>
<td>98</td>
<td>167</td>
<td>5</td>
<td>243</td>
<td>192</td>
<td>199</td>
<td>115</td>
<td>140</td>
<td>152</td>
<td>147</td>
<td>43</td>
<td>217</td>
<td>188</td>
<td>76</td>
<td>130</td>
</tr>
<tr>
<td>30</td>
<td>155</td>
<td>87</td>
<td>60</td>
<td>253</td>
<td>212</td>
<td>224</td>
<td>22</td>
<td>103</td>
<td>66</td>
<td>111</td>
<td>24</td>
<td>138</td>
<td>23</td>
<td>229</td>
</tr>
<tr>
<td>190</td>
<td>78</td>
<td>196</td>
<td>214</td>
<td>218</td>
<td>158</td>
<td>222</td>
<td>73</td>
<td>160</td>
<td>251</td>
<td>245</td>
<td>142</td>
<td>187</td>
<td>47</td>
<td>238</td>
</tr>
<tr>
<td>169</td>
<td>104</td>
<td>121</td>
<td>145</td>
<td>21</td>
<td>178</td>
<td>7</td>
<td>63</td>
<td>148</td>
<td>194</td>
<td>16</td>
<td>137</td>
<td>11</td>
<td>34</td>
<td>95</td>
</tr>
<tr>
<td>128</td>
<td>127</td>
<td>93</td>
<td>154</td>
<td>90</td>
<td>144</td>
<td>50</td>
<td>39</td>
<td>53</td>
<td>62</td>
<td>204</td>
<td>231</td>
<td>191</td>
<td>247</td>
<td>151</td>
</tr>
<tr>
<td>255</td>
<td>25</td>
<td>48</td>
<td>179</td>
<td>72</td>
<td>165</td>
<td>181</td>
<td>209</td>
<td>215</td>
<td>94</td>
<td>146</td>
<td>42</td>
<td>172</td>
<td>86</td>
<td>170</td>
</tr>
<tr>
<td>79</td>
<td>184</td>
<td>56</td>
<td>210</td>
<td>150</td>
<td>164</td>
<td>125</td>
<td>182</td>
<td>118</td>
<td>252</td>
<td>107</td>
<td>226</td>
<td>156</td>
<td>116</td>
<td>4</td>
</tr>
<tr>
<td>69</td>
<td>157</td>
<td>112</td>
<td>89</td>
<td>100</td>
<td>113</td>
<td>135</td>
<td>32</td>
<td>134</td>
<td>91</td>
<td>207</td>
<td>101</td>
<td>230</td>
<td>45</td>
<td>168</td>
</tr>
<tr>
<td>27</td>
<td>96</td>
<td>37</td>
<td>173</td>
<td>174</td>
<td>176</td>
<td>185</td>
<td>246</td>
<td>28</td>
<td>70</td>
<td>97</td>
<td>105</td>
<td>52</td>
<td>64</td>
<td>126</td>
</tr>
<tr>
<td>85</td>
<td>71</td>
<td>163</td>
<td>35</td>
<td>221</td>
<td>81</td>
<td>175</td>
<td>58</td>
<td>195</td>
<td>92</td>
<td>249</td>
<td>206</td>
<td>186</td>
<td>197</td>
<td>234</td>
</tr>
<tr>
<td>44</td>
<td>83</td>
<td>13</td>
<td>110</td>
<td>132</td>
<td>40</td>
<td>132</td>
<td>9</td>
<td>211</td>
<td>223</td>
<td>205</td>
<td>244</td>
<td>65</td>
<td>129</td>
<td>77</td>
</tr>
<tr>
<td>106</td>
<td>220</td>
<td>55</td>
<td>200</td>
<td>108</td>
<td>193</td>
<td>171</td>
<td>250</td>
<td>36</td>
<td>225</td>
<td>123</td>
<td>8</td>
<td>12</td>
<td>189</td>
<td>177</td>
</tr>
<tr>
<td>120</td>
<td>136</td>
<td>149</td>
<td>139</td>
<td>227</td>
<td>99</td>
<td>232</td>
<td>109</td>
<td>233</td>
<td>203</td>
<td>213</td>
<td>254</td>
<td>59</td>
<td>0</td>
<td>29</td>
</tr>
<tr>
<td>242</td>
<td>239</td>
<td>183</td>
<td>14</td>
<td>102</td>
<td>88</td>
<td>208</td>
<td>228</td>
<td>166</td>
<td>119</td>
<td>114</td>
<td>248</td>
<td>235</td>
<td>117</td>
<td>75</td>
</tr>
<tr>
<td>49</td>
<td>68</td>
<td>80</td>
<td>180</td>
<td>143</td>
<td>237</td>
<td>31</td>
<td>26</td>
<td>219</td>
<td>153</td>
<td>141</td>
<td>51</td>
<td>159</td>
<td>17</td>
<td>131</td>
</tr>
</tbody>
</table>
Final Pass: Producing Digest

[Diagram of the process of producing a digest from a padded message with appended checksum, including steps involving MD intermediate, message block, substitution, and final MD2 after checksum processed.]
MD4, MD5, SHA-1

- Input: arbitrary number of bits
- Output: 128 bits for MD4 and MD5, 160 bits for SHA-1
- Step 1: Pad the message to be a multiple of 512 bits (16 words, 64 octets)
- Step 2: Process the message, 512 bits at a time, to produce the message digest
Step 1: Padding

Padding for MD4, MD5, SHA-1
Step 2: Producing Message Digest

Overview of MD4, MD5, SHA-1
SHA-1 vs. MD5

- SHA-1 has 160-bit digest and MD5 128-bit, making SHA-1 more secure against brute-force.
- SHA-1 involves more stages and bigger buffer, and thus executes more slowly than MD5.
- MD5 is considered broken in 2004.
- SHA-1 is considered broken in 2005.
- SHA-2: four digest sizes 224, 256, 384, 512 bits.
Problem With Keyed Hash

- Keyed hash: $h(key|m)$
- An attacker gets $m$ and $h(key|m)$
- First pads $m$ according to the used hash function, and then adds another message $m'$ at the end, the result is $m|pad|m'$
- $h(key|m|pad|m')$ can be calculated from $h(key|m|pad)$ which is the intermediate digest
Solutions

- Use $h(m|key)$
- Use only half the bits of $h(key|m)$
- Use $h(key|m|key)$
- HMAC
Reading Assignments

- [Kaufman] Chapter 5