Since last time…
SHA-1 Collision

Expected behavior: different hashes

Doc 1
Sha-1
42C1..21
Good doc

Doc 2
Sha-1
3E2A..AE
Bad doc

Collision attack: same hashes

Sha-1

3713..42

Overview: SHA-1 collision occurs when two different files produce the same hash value, compromising data integrity.
CloudBleed

#cloudbleed
CloudPets
On to Key Management…
Key Management

- key generation
- key exchange
- key storage
- key use
- key replacement
## What sort of keys?

<table>
<thead>
<tr>
<th>Security</th>
<th>Symmetric</th>
<th>FFC (e.g., DSA, DH)</th>
<th>IFC (e.g., RSA)</th>
<th>ECC (e.g., ECDSA)</th>
<th>DS, hash</th>
<th>HMAC, PRG</th>
<th>NIST Rec.</th>
</tr>
</thead>
<tbody>
<tr>
<td>≤ 80</td>
<td>2TDEA</td>
<td>(1024,160)</td>
<td>k = 1024</td>
<td>f = 160-223</td>
<td>SHA-1</td>
<td></td>
<td>No</td>
</tr>
<tr>
<td>112</td>
<td>3TDEA</td>
<td>(2048, 224)</td>
<td>k = 2048</td>
<td>f = 224-255</td>
<td>SHA-224</td>
<td></td>
<td>until 2030</td>
</tr>
<tr>
<td>128</td>
<td>AES-128</td>
<td>(3072, 256)</td>
<td>k = 3072</td>
<td>f = 256-383</td>
<td>SHA-256</td>
<td>SHA-1</td>
<td>Yes</td>
</tr>
<tr>
<td>≥ 256</td>
<td>AES-256</td>
<td>(15360, 512)</td>
<td>k = 15360</td>
<td>f = 512+</td>
<td>SHA-512</td>
<td>SHA-256</td>
<td>Yes</td>
</tr>
</tbody>
</table>
Failure cases
Generating secure keys

- /dev/random
- /dev/urandom
Failure cases

```c
int getRandomNumber()
{
    return 4;  // chosen by fair dice roll.
    // guaranteed to be random.
}
```
Key Storage

- cryptographic module
- remote storage
Secret Sharing
Key use

Don’t reuse keys
## Key Replacement

<table>
<thead>
<tr>
<th>Key Type</th>
<th>Cryptoperiod</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Originator (OUP)</td>
</tr>
<tr>
<td>Private Signature Key</td>
<td>1 to 3 years</td>
</tr>
<tr>
<td>Public Signature-Verification Key</td>
<td>Several years (depends on key size)</td>
</tr>
<tr>
<td>Symmetric Authentication Key</td>
<td>≤ 2 years</td>
</tr>
<tr>
<td>Private Authentication Key</td>
<td>1 to 2 years</td>
</tr>
<tr>
<td>Public Authentication Key</td>
<td>1 to 2 years</td>
</tr>
<tr>
<td>Symmetric Data Encryption Keys</td>
<td>≤ 2 years</td>
</tr>
<tr>
<td>Symmetric Key Wrapping Key</td>
<td>≤ 2 years</td>
</tr>
<tr>
<td>Symmetric Master Key</td>
<td>About 1 year</td>
</tr>
<tr>
<td>Private Key Transport Key</td>
<td></td>
</tr>
<tr>
<td>Public Key Transport Key</td>
<td>1 to 2 years</td>
</tr>
<tr>
<td>Symmetric Authorization Key</td>
<td></td>
</tr>
<tr>
<td>Private Authorization Key</td>
<td></td>
</tr>
<tr>
<td>Public Authorization Key</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Password-Based Encryption

Through 20 years of effort, we’ve successfully trained everyone to use passwords that are hard for humans to remember, but easy for computers to guess.
Fixing Password-Based Encryption

- slow hash functions
- salt
PBKDF2

\[ P \rightarrow \text{PRF} \rightarrow \cdots \rightarrow T_1 \]

\[ P \rightarrow \text{PRF} \rightarrow \cdots \rightarrow T_2 \]

\[ \ldots \]

\[ P \rightarrow \text{PRF} \rightarrow \cdots \rightarrow T_{\text{len}} \]

\[ MK = T_1 || \ldots || T_{\text{len}} < \text{len} > \]
PBE in practice
ID-based encryption

- use ID as public key
- key generation authority generates secret keys for each ID
- Generalization: attribute-based encryption