Digital Signatures for Flows and Multicasts

by Chung Kei Wong and Simon S. Lam

in IEEE/ACM Transactions on Networking, August 1999

Digital Signature

- Examples: RSA, DSA
- Provide authenticity, integrity and non-repudiation
- How to sign/verify?
  - signing key $k_s$, verification key $k_v$, message digest $h(m)$
  - $\text{signature} = \text{sign}(h(m), k_s)$
  - $\text{verify}(\text{signature}, h(m), k_v) = \text{True}/\text{False}$
- Signing & verification operations are slow compared to symmetric key operations
Motivation

- Traditional network applications (circa 1998)
  - message-oriented unicast, e.g., email, file transfer, client-server
- Emerging network applications
  - flow-oriented, e.g., audio, video, stock quotes
  - multicast, e.g., teleconference, software distribution
- Problem: How to sign efficiently?
  - high-speed transmissions
  - real-time generated flows
  - delay-sensitive packet flows

All-or-nothing flows

- The signer generates a message digest of the entire flow (file) and signs the message digest
- But most Internet applications do not create all-or-nothing flows
  - a flow is sent as a sequence of packets
  - each packet is used as soon as it is received
**Sign-each Approach**

- A flow is a sequence of data packets
- Sign each packet individually
- Inefficient: one signing/verification operation per packet
- Rates on a Pentium-II 300 MHz using 100% processing time (with 512-bit modulus)

<table>
<thead>
<tr>
<th>Packet size (bytes)</th>
<th>Signing Rate (packets/sec)</th>
<th>Verification Rate (packets/sec)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>RSA</td>
<td>DSA</td>
</tr>
<tr>
<td>512</td>
<td>78.8</td>
<td>176</td>
</tr>
<tr>
<td>1024</td>
<td>78.7</td>
<td>175</td>
</tr>
</tbody>
</table>

**Prior work on signing digital streams**

- [Gennaro and Rohatgi 1997]
- One signing/verification op for an entire flow—only the first packet is signed
  - Each packet contains authentication info for next
- Verification of each packet depends on previous ones
  - Reliable delivery required
Flow Signing Problem

- Each packet may be used as soon as it is received
- Subsequences of a flow are received and used
  - best-effort delivery, e.g., UDP, IP multicast
  - different needs/capabilities, e.g., layered video
- How to efficiently sign flows with each packet being *individually verifiable*?

Our Approach: Chaining

- Partition a flow into blocks of packets
  - Sign the digest of each block instead of each packet individually
- Each packet carries its own authentication information to prove it is in the block
  - Authentication info provided by chaining
**Star Chaining – Signing**

Block digest $D_{1-8} = h(D_1, \ldots, D_8)$

- **Block signature** = $\text{sign}(D_{1-8})$
- **Packet signature for packet $P_3$**:
  - $\text{sign}(D_{1-8}), D_1, D_2, D_4, \ldots, D_8$
- *Chaining overhead is $\mathcal{O}(\text{block size})$*

**Star Chaining – Verification**

- **Verifying first received packet (say $P_3$)**
  - Block digest $D'_{1-8} = h(D_1, D_2, D'_3, D_4, \ldots, D_8)$

  - $\text{verify}(D'_{1-8}, \text{sign}(D_{1-8}))$

- **Caching of verified nodes**
  - no verification op for other packets in the block

Digital Signatures (Simon Lam)
Tree Chaining – Signing

- [Merkle 1989]
- Block signature = \( \text{sign}(D_{1-8}) \)
- Packet signature for packet \( P_3 \):
  \( \text{sign}(D_{1-8}), D_4, D_{1-2}, D_{5-8} \)
- Chaining overhead is \( \mathcal{O}(\log(\text{block size})) \)

Tree Chaining – Verification

- Verifying first received packet (say \( P_3 \))
  \( \text{verify}(D'_{1-8}, \text{sign}(D_{1-8})) \)
- Caching of verified nodes
  \( \text{no verification op for other packets in the block} \)

Block digest \( D'_{1-8} = h(D'_{1-4}, D_{5-8}) \)

Packet digests
**Chaining Technique: Signer Overhead**

1. **Compute packet digests**
   - Digest comp time
2. **Build authentication tree**
   - Tree build time
3. **Sign block digest**
   - Signature comp time
4. **Build packet signatures**
   - Packet signature build time

Chaining time = Tree build time + Packet signature build time

**Chaining Technique: Verifier Overhead**

1. **Build authentication tree**
   - Tree build time
2. **Compute packet digests**
   - Digest comp time
3. **Verify chaining information**
   - Chaining verification time
4. **Verify block signature**
   - Signature verifying time

Chaining time = Tree build time + Chaining verification time
**Chaining Time Overheads**

- **at sender**
  - Overheads increase linearly with block size (in log scale)
  - Much smaller than signing/verification times

- **at receiver**

**Chaining Overhead Size**

- Smallest when tree degree is 2
- Increases linearly with logarithm of block size
- Packet signature = block signature + chaining overhead

---

Digital Signatures (Simon Lam) 15

Digital Signatures (Simon Lam) 16
Flow Signing/Verification Rates

- 1024-byte packets, RSA with 512-bit modulus
- Increases with block size
- Varies only slightly with tree degree
  - we recommend degree 2 tree chaining

Digital Signatures (Simon Lam)  

Flow Signing/Verification Rates

- Degree two tree, RSA with 512-bit modulus, three different packet sizes

Digital Signatures (Simon Lam)
Real-time Generated Flows

- Fixed block size for non-real-time generated flows
- Fixed time period $T$ for real-time generated flows
- Bounded delay signing since for any packet
  \[ \text{delay} \leq T + T_{\text{chain}} + T_{\text{sign}} \]
  \[
  \begin{array}{c}
  \text{period T} \quad \text{period T} \\
  m_1 \text{ packets} \quad m_2 \text{ packets}
  \end{array}
  \]

- $T$ should be larger than $T_{\text{chain}} + T_{\text{sign}}$
- delay cannot be smaller than $2(T_{\text{chain}} + T_{\text{sign}})$

Selecting a Signature Scheme

- RSA: signing rate not high enough
- DSA: both rates not high and verification rate < signing rate
  - In a group, receivers may have widely different resources, e.g., PDAs, notebooks, desktops

- We proposed several extensions to FFS
  [Feige, Fiat and Shamir 1986]
**FFS Signer**

- Choose two large primes $p$ and $q$
- Compute modulus $n = pq$
- Choose integers $v_1, \ldots, v_k$
  
  $s_1, \ldots, s_k$
  
  such that $s_l^2 = v_l^{-1} \mod n$
- Signing key is $\{s_1, \ldots, s_k, n\}$
- Verification key is $\{v_1, \ldots, v_k, n\}$

---

**How to Sign Message $m$**

- Choose $t$ random integers, $r_1, \ldots, r_t$, between 1 and $n$
- Compute $x_i = r_i^2 \mod n$, for $i = 1, \ldots, t$
- Compute message digest $h(m, x_1, \ldots, x_t)$
  
  where function $h(\cdot)$ is public knowledge and
  produces a digest of at least $k \times t$ bits
  
  let $\{b_{ij}\}$ be the first $k \times t$ bits of the digest
- Compute $y_i = r_i x (s_1^{b_{i1}} x \ldots x s_k^{b_{ik}}) \mod n$
  
  for $i = 1, \ldots, t$
- Signature of $m$ consists of
  
  $\{y_i\}$ and $\{b_{ij}\}$ for $i = 1, \ldots, t$ and $j = 1, \ldots, k$
How to Verify Signature of Message m

- signature of \( m \)
  \( \{y_i\} \) and \( \{b_{ij}\} \) for \( i = 1, \ldots, t \) and \( j = 1, \ldots, k \)
- compute \( z_i = y_i^2 \times (v_1^{b_{i1}} \times \ldots \times v_k^{b_{ik}}) \mod n \)
  for \( i = 1, \ldots, t \)
  it can be shown that \( z_i \) is equal to \( x_i \) at the signer
- signature is valid if and only if the first \( k \times t \) bits of \( h(m, z_1, \ldots, z_t) \) are equal to the \( \{b_{ij}\} \)
  received in signature

FFS(k,t)

- security level increases with
  - size of modulus \( n \) (or size of primes \( p \) and \( q \))
  - value of product \( kt \)
- key size is \( (k+1) \times |n| \)
  assuming \( |n| = |v_i| \) or \( |s_i| \) in bits
- signature size is \( t \times |n| + k \times t \) bits
  minimized for \( t=1 \)
**FFS key and signature sizes**

**FFS Signing/Verification Key and Signature Sizes (Bytes) With 512-Bit Modulus**

<table>
<thead>
<tr>
<th></th>
<th>$t = 1$</th>
<th>$t = 2$</th>
<th>$t = 4$</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>key</td>
<td>sig</td>
<td>key</td>
</tr>
<tr>
<td>$kt = 64$</td>
<td>4160</td>
<td>72</td>
<td>2112</td>
</tr>
<tr>
<td>$kt = 128$</td>
<td>8256</td>
<td>80</td>
<td>4160</td>
</tr>
</tbody>
</table>

For a fixed $kt$ product, signature size is minimized for $t=1$, but key size is maximized.

---

**eFFS Signature Scheme**

- Several extensions to FFS [Feige, Fiat and Shamir 1986]
  - Faster signing
    - Chinese remainder theorem (*crt*)
    - Precomputation (*4-bit, 8-bit*)
  - Faster verification
    - Small verification key (*sv-key*) [Micali & Shamir 1990]
  - Adjustable and incremental verification
    - multilevel signature
    - lower security level with less processor time at receiver
    - security level can be increased later by more processor time
**eFFS extension (1)**

- **Chinese remainder theorem**
  instead of \( y_i = r_i \times (s_1^{b_{li}} \times ... \times s_k^{b_{ki}}) \mod n \)
  signer computes
  \[
  a_i = r_i \times (s_1^{b_{li}} \times ... \times s_k^{b_{ki}}) \mod p \\
  b_i = r_i \times (s_1^{b_{li}} \times ... \times s_k^{b_{ki}}) \mod q \\
  y_i = ((a_i - b_i) \times q \times q_p^{-1} + b_i) \mod n
  \]
  where \( q_p^{-1} \) denotes \( q^{-1} \mod p \),
  - multiplications in mod \( p \) and mod \( q \) faster than in mod \( n \)

- **Only signer knows \( p \) and \( q \)**

---

**eFFS extension (2)**

- **small verification key** [Micali & Shamir]:
  use first \( k \) prime numbers that satisfy
  \[
  s^2 = p^{-1} \mod n
  \]
  where \( p \) is prime and \( s \) is an integer

- **faster verifying time and smaller key size**
eFFS extension (3)

- To compute \( y_i = r_i \times (s_1^{b_1} \times \ldots \times s_k^{b_k}) \mod n \) for \( i = 1, \ldots, t \)

- Precomputation of \( (s_1^{b_1} \times \ldots \times s_k^{b_k}) \)

  Additional memory of 31 KB and 261 KB required for 4-bit and 8-bit precomp respectively

  Only minor improvement at verifier when used with small v-key

---

**eFFS - Signing**

- sv-key does not reduce signing time
- crt reduces signing time by 10-20%
- 8-bit + crt reduces signing time by 60-70%
**eFFS - Verification**

- sv-key reduces verification time by 90%
- 4-bit or 8-bit slightly reduces verification time

**eFFS Key Size**

- Large signing key 8000-17000 bytes
  - private to signer
- Verification key 300-400 bytes
eFFS Signature Size

- Signature size comparable to RSA and Rabin

Signing Time Comparison

- 8-bit + crt + sv-key extensions
- eFFS has the smallest signing time
Verification Time Comparison

- DSA and ElGamal verification times very large
- Rabin, RSA and eFFS too small to see

eFFS verification time comparable to RSA (Rabin most efficient verification)
Flow Signing/Verification Rates

- 1024-byte packets, block size 16, degree two tree chaining
- eFFS has highest signing rate
- eFFS verification rate comparable to RSA

Digital Signatures (Simon Lam) 37

---

eFFS Adjustable and Incremental Verification

- Security level of eFFS($k, t$) depends on modulus size and product $kt$
  - same $kt$ and modulus size $\sim$ same security level

- Adjustable and incremental verification
  - using $t > 1$ with additional info in signature
  - up to $t$ steps
  - adjustable and incremental:
    receiver verifies steps one by one

Digital Signatures (Simon Lam) 38
**eFFS Adjustable and Incremental Verification (cont.)**

- $t$-level signature includes $\{x_i\}$ for $i = 2, \ldots, t$
  
  note that $\{x_i\}$ can be computed from original signature together with verification key

- verify a $t$-level signature at security level $l \leq t$,
  
  1. compute $z_i = y_i^2 \times (v_1^{b_1} \times \ldots \times v_k^{b_k}) \mod n$ for $i = 1, \ldots, l$,
  2. verify that the first $k \times t$ bits of $h(m, z_1, x_2, \ldots, x_t)$ are equal to the $\{b_{ij}\}$ received, and $z_2, \ldots, z_l$ are equal to $x_2, \ldots, x_l$

---

**eFFS Adjustable and Incremental Verification (cont.)**

- increase security level from $l_1$ to $l_2$,
  
  1. compute $z_i = y_i^2 \times (v_1^{b_1} \times \ldots \times v_k^{b_k}) \mod n$ for $i = l_1 + 1, \ldots, l_2$,
  2. verify that $z_{l_1+1}, \ldots, z_{l_2}$ are equal to $x_{l_1+1}, \ldots, x_{l_2}$
Incremental signing times

cFFS \( t \)-LEVEL SIGNATURE SIGNING TIMES (MILLISECONDS)

<table>
<thead>
<tr>
<th></th>
<th>( kt = 32 )</th>
<th>( kt = 64 )</th>
<th>( kt = 128 )</th>
</tr>
</thead>
<tbody>
<tr>
<td>1-level signature</td>
<td>1.47</td>
<td>2.02</td>
<td>3.14</td>
</tr>
<tr>
<td>2-level signature</td>
<td>2.87</td>
<td>3.98</td>
<td>5.67</td>
</tr>
</tbody>
</table>

2-level signature takes less time to sign than two 1-level signatures

Incremental verification times

cFFS INCREMENTAL VERIFICATION TIMES (MILLISECONDS) FOR \( kt = 128 \).

(a) 2-LEVEL SIGNATURE. (b) 4-LEVEL SIGNATURE.

<table>
<thead>
<tr>
<th></th>
<th>To</th>
<th>level 1</th>
<th>level 2</th>
</tr>
</thead>
<tbody>
<tr>
<td>From level 0</td>
<td>0.42</td>
<td>0.81</td>
<td></td>
</tr>
<tr>
<td>From level 1</td>
<td>0.40</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

(a)

<table>
<thead>
<tr>
<th></th>
<th>To</th>
<th>level 1</th>
<th>level 2</th>
<th>level 3</th>
<th>level 4</th>
</tr>
</thead>
<tbody>
<tr>
<td>From level 0</td>
<td>0.34</td>
<td>0.63</td>
<td>0.03</td>
<td>1.22</td>
<td></td>
</tr>
<tr>
<td>From level 1</td>
<td>0.30</td>
<td>0.60</td>
<td>0.89</td>
<td></td>
<td></td>
</tr>
<tr>
<td>From level 2</td>
<td>0.30</td>
<td>0.60</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>From level 3</td>
<td>0.31</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

(b)
Conclusions

- **Flow signing/verification procedures**
  - much more efficient than sign-each
  - small communication overhead
  - can be used by a sender that signs a large number of packets to different receivers
    - there is no requirement that the packets belong to a flow but if they do, verification is also more efficient

- **eFFS digital signature scheme**
  - most efficient signing compared to RSA, Rabin, DSA, and ElGamal
  - highly efficient verification and comparable to RSA (only Rabin is more efficient)
  - adjustable and incremental verification

End