

Hash Function Firewalls in Signature Schemes

Burt Kaliski, RSA Laboratories
IEEE P1363 Working Group Meeting
June 2, 2000
(Rev. June 8, 2000)



Outline

- Hash function flexibility and firewalls
- Breaking firewalls in signature schemes
- Conclusions



Hash Function Flexibility

- Many signature schemes allow multiple hash functions, to enable improvement over time
- Signer typically selects from a small set
- Verifier may accept a larger set, for interoperability with many signers



Weak Hash Function Risks

- Signer accepts the risk that a hash function in its set may turn out to be weak
 - possibly enabling an attacker to forge signatures
- However, signer may also be at risk if a hash function in the *verifier's* set is weak
- Signer accepts the risk from its own choices, but needs some way to mitigate the risk due to the verifier's choices



Mitigating the Risk

- One approach is to limit the verifier's set to "trusted" hash functions
 - only SHA-1 in FIPS 186
 - only "ANSI-approved"
- Another approach is for the signer to indicate acceptable hash functions in its certificate
- Alternatively, the signature scheme itself might somehow distinguish between different hash functions: a "firewall"



Hash Function Firewalls

- Apparently first suggested by J. Linn in development of Privacy-Enhanced Mail, ca. 1990
- RSA signature on message M :
 1. Let $f = \text{Pad} || \text{HashID} || \text{Hash}(M)$
 2. Apply RSA signature primitive to f
- HashID is a "firewall" against weak hash functions
 - protected directly by signature primitive
 - existing signatures cannot be reused with a different, weak hash function



Does It Work?

- Hash function firewalls have become a common practice in many signature schemes and standards
- A firewall prevents an attacker from reusing an *existing signature* with a different hash function
- But what about other kinds of signature forgery?



Summary of Results

- Firewalls in several signature schemes *do not protect* against signature forgery with a weak hash function
- If attacker can invert a hash function in these schemes, attacker can forge a signature
- Signer doesn't need to support the weak hash function — doesn't even need to be involved!
 - same concept as presented by Brown and Johnson at the March 2000 P1363 meeting w.r.t. PV signatures — but extended to address other signature schemes and hash function firewalls



Outline of Attacks

- ISO/IEC 9796-2
- GQ from ISO/IEC 14888-2
 - may extend to other schemes based on proofs of knowledge
- DSA with hash ID
 - extends to ECDSA
- PSS-R from IEEE P1363a D3



General Approach

- Attacker's goal is twofold:
 1. Develop a signature that identifies a *weak* hash function that the verifier accepts
 2. Find a message M with the correct hash under the weak hash function
- Notation:
 - WeakHash: weak hash function
 - WeakHashID: identifier for weak hash function
 - M_r : recoverable message part
 - M_{nr} : nonrecoverable message part
 - M : message



ISO 9796-2 Scheme

- Signature scheme with message recovery based on integer factorization problem
- Recommendations for addressing weak hash function attacks:
 - “1. Require a particular hash function ...
 - “2. Allow a set of hash functions and explicitly indicate in every signature the hash-function in use by an identifier included as part of the signature calculation ...”
- HashID is a one-byte ID from ISO 10118



ISO 9796-2 with a Firewall

- Public key: modulus n , exponent e
- Private key: $d = e^{-1} \bmod \phi(n)$
 - RSA version; RW version is similar
- Sign(M_r, M_{nr}) = s :
 1. Let $T = \text{Hash}(M_r \parallel M_{nr})$
 2. Let $f = 6b \text{ bb } \dots \text{ bb ba} \parallel M_r \parallel T \parallel \text{HashID} \parallel \text{cc}$
 3. Let $s = f^e \bmod n$
- Recover(M_r, s) = M_r :
 1. Let $f = s^e \bmod n$
 2. Decode $f = 6b \text{ bb } \dots \text{ bb ba} \parallel M_r \parallel T \parallel \text{HashID} \parallel \text{cc}$
 3. Check $T = \text{Hash}(M_r \parallel M_{nr})$



Does the Firewall Work?

- HashID prevents an attacker from reusing an existing signature with a different hash function
- But it doesn't prevent an attacker from forging a new signature with a weak hash function



Breaking the Firewall

1. Select s in $[1, n-1]$
 2. Let $f = s^e \bmod n$
 3. Decode $f = 6b\ bb \dots bb\ ba \parallel M_r \parallel T \parallel \text{HashID} \parallel cc$
 4. If decode error or $\text{HashID} \neq \text{WeakHashID}$, goto 1
 5. Solve for M_{nr} such that $T = \text{WeakHash}(M_r \parallel M_{nr})$
 6. Output M_r, M_{nr}, s
- Expect $\sim 2^{24}$ tries to get desired hash ID, padding



GQ Scheme from ISO/IEC 14888-2

- Signature scheme with appendix based on discrete logarithm problem
- Recommendations for addressing weak hash function attacks:
 - “The hash-function identifier shall be included in the hash-token unless the hash-function is uniquely determined by the signature mechanism or by the domain parameters”
- HashID is a one-byte ID from ISO 10118



GQ Signatures with a Firewall

- Domain parameters: modulus N , exponent V
- Public key: Y (identity-based)
- Private key: $X = Y^{1/V} \bmod N$
- Sign $(M) = (R, S)$
 1. Let $\Pi = K^V \bmod N$, K random
 2. Let $R = \text{Hash}(\Pi \parallel M) \parallel \text{HashID}$
 3. Let $S = KX^R \bmod N$
- Verify $(M, (R, S))$:
 1. Let $\Pi = Y^R S^V \bmod N$
 2. Check $R = \text{Hash}(\Pi \parallel M) \parallel \text{HashID}$



Breaking the Firewall

1. Select S in $[1, n-1]$, hash target T
 2. Let $R = T \parallel \text{WeakHashID}$
 3. Let $\Pi = Y^R S^V \bmod N$
 4. Solve for M such that $T = \text{WeakHash}(\Pi \parallel M)$
 5. Output $M, (R, S)$
- Only *one* try
 - Target is prespecified, which may simplify inversion



DSA Scheme

- Signature scheme with appendix based on discrete logarithm problem
- In FIPS 186, a unique hash function: SHA-1
- However, P1363 allows hash function flexibility
- Consider a variation of DSA with a firewall



DSA Signatures with a Firewall

- Domain parameters: prime p , base g , order q
- Public key $y = g^x \bmod p$
- Private key x
- Sign $(M) = (r, s)$:
 1. Let $r = (g^k \bmod p) \bmod q$, k random
 2. Let $R = \text{Hash}(M) \parallel \text{HashID}$
 3. Let $s = k^{-1} (R + xr) \bmod q$
- Verify $(M, (r, s))$:
 1. Let $R = \text{Hash}(M) \parallel \text{HashID}$
 2. Let $a = Rs^{-1} \bmod q$, $b = rs^{-1} \bmod q$
 3. Check $r = (g^a y^b \bmod p) \bmod q$



Breaking the Firewall

1. Select a, b in $[1, q-1]$
 2. Let $r = (g^a y^b \bmod p) \bmod q$
 3. Let $s = rb^{-1} \bmod q$, $R = as \bmod q$
 4. Decode $R = T \parallel \text{HashID}$
 5. If $\text{HashID} \neq \text{WeakHashID}$, goto 1
 6. Solve for M such that $T = \text{WeakHash}(M)$
 7. Output $M, (r, s)$
- Expect ~256 tries for minimum q



PSS-R from IEEE P1363a D3

- Signature scheme with message recovery based on integer factorization problem
- As drafted, a hash function firewall based on ISO 10118



PSS-R Signatures (D3 version)

- Public key: modulus n , exponent e
- Private key: $d = e^{-1} \bmod \phi(n)$
- Sign $(M_r, M_{nr}) = s$:
 1. Let $T = \text{Hash}(\text{salt} \parallel \text{len}(M_r) \parallel \text{Hash}(M_r \parallel M_{nr}))$
 2. Let $U = G(T) \oplus (\text{salt} \parallel 00 \dots 01 \parallel M_r)$
 3. Let $f = 6b \parallel T \parallel U \parallel \text{HashID} \parallel \text{cc}$
 4. Let $s = f^d \bmod n$
- Recover $(M_{nr}, s) = M_r$:
 1. Let $f = s^e \bmod n$
 2. Decode $f = 6b \parallel T \parallel U \parallel \text{HashID} \parallel \text{cc}$
 3. Decode $U \oplus G(T) = (\text{salt} \parallel 00 \dots 01 \parallel M_r)$
 4. Check $T = \text{Hash}(\text{salt} \parallel \text{len}(M_r) \parallel \text{Hash}(M_r \parallel M_{nr}))$



Breaking the Firewall

1. Select s in $[1, n-1]$
 2. Let $f = s^e \bmod n$
 3. Decode $f = 6b \parallel T \parallel U \parallel \text{HashID} \parallel \text{cc}$
 4. Decode $U \oplus G(T) = (\text{salt} \parallel 00 \dots 01 \parallel M_r)$
 5. If decode error or $\text{HashID} \neq \text{WeakHashID}$, goto 1
 6. Solve for M_{nr} such that
 $T = \text{WeakHash}(\text{salt} \parallel \text{len}(M_r) \parallel \text{WeakHash}(M_r \parallel M_{nr}))$
 7. Output M_r, M_{nr}, s
- Expect ~2³² tries



Comparison of Attacks

- Decreasing difficulty for attacker:
 - PSS-R: ~2³² tries, M_r constrained
 - ISO 9796-2: ~2²⁴ tries, M_r constrained
 - DSA: ~256 tries, none of message constrained
 - GQ: one try, hash target specified by attacker
- None of the attacks involves the actual signer — all can be performed with access only to the signer's public key



Other Schemes with Firewalls

- ISO/IEC 9796-4 (= DL/ECSSR in IEEE P1363a D4)
 - optional firewall, can be broken
- ANSI X9.31
 - firewall is protected if minimum amount of padding, met in typical use
- PKCS #1 v1.5
 - firewall is protected by minimum amount of padding
- PSS and PSS-R in IEEE P1363a D4
 - no hash ID firewall, yet still protected if minimum amount of padding and G function is based on underlying hash function, except for “pathological” cases



PSS-R Signatures (D4 version)

- $\text{Sign}(M_r, M_{nr}) = s$:
 1. Let $T = \text{Hash}(\text{len}(M_r) || M_r || \text{Hash}(M_{nr}) || \text{salt})$
 2. Let $U = G(T) \oplus (00 \dots 01 || M_r || \text{salt})$
 3. Let $f = U || T || \text{bc}$
 4. Let $s = f^d \text{ mod } n$
- $\text{Recover}(M_{nr}, s) = M_r$:
 1. Let $f = s^e \text{ mod } n$
 2. Decode $f = U || T || \text{bc}$
 3. Decode $U \oplus G(T) = (00 \dots 01 || M_r || \text{salt})$
 4. Check $T = \text{Hash}(\text{len}(M_r) || M_r || \text{Hash}(M_{nr}) || \text{salt})$



Firewall Without a Hash ID

- With two requirements, the current PSS-R can heuristically protect against weak hash function attacks:
 - G must be based on the underlying hash function
 - $(00 \dots 01 || M_r || \text{salt})$ must have a minimum amount of padding
- These requirements protect against weaknesses in conventional hash functions
 - only “pathological” hash functions are a risk, and these are unlikely to be accepted by a verifier
- Padding requirement met by PSS with appendix (i.e., M_r empty) for typical hash function sizes



Firewall Protection

- Attacker wants $f = s^e$ that can be decoded as

$$f = U || T || \text{bc}$$

$$U \oplus G(T) = (00 \dots 01 || M_r || \text{salt})$$
- Existing signatures cannot be reused with a weak hash function because $G(T)$ will be different
- New signatures cannot be forged because $U \oplus G(T)$ will be unlikely to have the minimum padding, for random f
 - inverting the hash function doesn't help
 - only attack is to coerce verifier to use a “pathological” hash function constructed to match the format



Schemes without Firewalls

- Full Domain Hashing
- Pintsov-Vanstone
- These were not intended to protect against weak hash functions, as originally specified
- But if they had a hash ID firewall, the firewall could be broken
 - for PV scheme, extension of attack on the non-firewall version (see D. Brown and D. Johnson, “Formal Security Proofs for a Signature Scheme with Partial Message Recovery”, <http://grouper.ieee.org/groups/1363/Research>)



Conclusions

- Hash function firewalls don't necessarily prevent weak hash function attacks!
- Scheme-specific analysis is needed
 - some schemes are protected, perhaps with a minimum amount of padding
 - some schemes are not
- Formal definitions of security against weak hash function attacks would be helpful
- In general, a verifier should be careful about the hash functions it accepts

