Digital Signatures and (Implicit) Certificates

Christopher A. Wood

University of California Irvine & Palo Alto Research Center

woodc1@uci.edu
cwood@parc.com

May 5, 2015

Overview

Motivating Signatures

Digital Signature Algorithms

Public Key Infrastructure

Implicit Certificates

Introduction

Scenario: Alice wants to send message m to Bob over a public channel subject to malicious adversaries

Bob receives message m' over the channel

Problem: How does Bob verify that $m = m'$?

Shared Key Solution

If Alice and Bob share a common key k...

Alice		Bob
$t := \text{HMAC}(k, m)$	$\xrightarrow{\text{m, t}}$	
		$t' = \text{HMAC}(k, m')$
		Accept m' if $t = t'$

HMAC is a standard Message Authentication Code algorithm – a keyed hash

Key Agreement

How do Alice and Bob agree on k?

- Carrier pidgeons?
- Diffie Hellman? (subject to Man-In-The-Middle)
- ...

Public-key cryptography is a “better” approach.

Public Key Solution

If Bob knows Alice's public key pk...

Alice		Bob
$\sigma := \mathsf{Sign}(sk, m)$		
	$\xrightarrow{\mathrm{m}, \sigma}$	
		$b = \mathsf{Verify}(pk, m', \sigma)$
		Accept m' if $b = 1$

Public Key Solution

If Bob knows Alice's public key *pk*...

Alice		Bob
$\sigma := \mathsf{Sign}(sk, m)$		
	$\xrightarrow{\mathrm{m}, \sigma}$	
		$b = \mathsf{Verify}(pk, m', \sigma)$
		Accept m' if $b = 1$

Question: What options do we have for Sign and Verify?

RSA Signatures

Private key d, public key e, RSA modulus N

Algorithm 1 Sign($sk = d, m$)

I $= H(m)$
$\sigma = I^d \mod N$
return σ

Algorithm 2 Verify($pk = e, m', \sigma$)

$I = \sigma^e \mod N$
$I' = H(m')$
return $I == I'$

ElGamal Signatures

Secret key x, public key $(p, g, y = g^x \mod p)$

Algorithm 3 Sign$(sk = x, m)$

$s = 0$
while $s = 0$ **do**
  $k \xleftarrow{\$} [2, p-2]$ such that $(k, p-1) = 1$
  $r := g^k \mod p$
  $s := (H(m) - xr)k^{-1} \mod (p-1)$
  if $s > 0$ **then**
    return (r, s)
  end if
end while

Algorithm 4 Verify$(pk = (p, g, y \equiv g^x \mod p), m', \sigma = (r, s))$

if not $(0 < r < p$ or $0 < s < p-1)$ **then**
  reject
end if
return $g^{H(m)} == y^r r^s$

DSA Sign

Public key $(p, q, g, y \equiv g^x \mod p)$, private key x

Algorithm 5 Sign$(sk = x, m)$

$r := 0; s := 0$
while $r == 0$ **do**
  $k \xleftarrow{\$} [1, q-1]$
  $r := (g^k \mod p) \mod q$
  if $r > 0$ **then**
    $s := k^{-1}(H(m) + xr) \mod q$
    if $s > 0$ **then**
      return $\sigma = (s, r)$
    end if
  end if
end while

DSA Verify

Public key $(p, q, g, y \equiv g^x \mod p)$, private key x

Algorithm 6 Verify$(pk = (p, q, g, y \equiv g^x \mod p), m', \sigma = (s, r))$

if not $(0 < r < q$ or $0 < s < q)$ **then**
    reject
end if
$w := s^{-1} \mod q$
$u_1 := H(m') \cdot w \mod q$
$u_2 := r \cdot w \mod q$
$v := ((g^{u_1} y^{u_2}) \mod p) \mod q$
return $v == r$

ECDSA Sign

Public parameters (E, G, n), private key q, public key $Q = qG$

Algorithm 7 Sign$(sk = q, m)$

$r := 0; s := 0$
$e := H(m)$
$z := L_n(e)$ {Leftmost n bits of e}
while $r == 0$ **do**
  $k \xleftarrow{\$} [1, n-1]$
  $(x_1, y_1) := kG$
  $r := x_1 \mod n$
  if $r > 0$ **then**
    $s := k^{-1}(z + rq) \mod n$
    if $s > 0$ **then**
      return $\sigma = (r, s)$
    end if
  end if
end while

ECDSA Verify

Public parameters (E, G, n), private key q, public key $Q = dG$

Algorithm 8 Verify$(pk = Q, m', \sigma = (r, s))$

if $r, s \notin [1, n-1]$ **then**
  reject
end if
$e := H(m')$
$z := L_n(e)$
$w := s^{-1} \mod n$
$u_1 := zw \mod n$
$u_2 := rw \mod n$
$(x_1, y_1) := u_1 G + u_2 Q$
return $r == x_1 \mod n$

Verification Improvements

Batch!

Public Key Infrastructure

Problem: How does Bob *obtain and trust* trust Alice's public key?

Public Key Infrastructure

Problem: How does Bob *obtain and trust* trust Alice's public key?

Solution: Public Key Infrastructure (PKI)

Public Key Infrastructure (cont'd)

PKI arrangement:

- Public keys and identities are bound together using **certificates**

Public Key Infrastructure (cont'd)

PKI arrangement:

- Public keys and identities are bound together using **certificates**
- Certificates are issued by certificate authorities (CAs)

Public Key Infrastructure (cont'd)

PKI arrangement:

- Public keys and identities are bound together using **certificates**
- Certificates are issued by certificate authorities (CAs)
- CAs are delegated permission to issue certificates by their parent CA

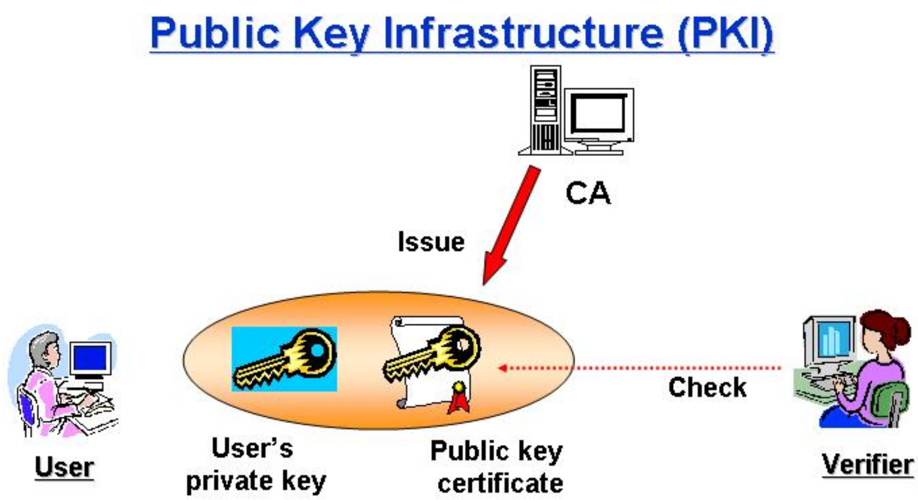
Public Key Infrastructure (cont'd)

PKI arrangement:

- Public keys and identities are bound together using **certificates**
- Certificates are issued by certificate authorities (CAs)
- CAs are delegated permission to issue certificates by their parent CA
- The root CA is "absolute" – a trusted anchor for the **certificate chain**

Key observation: Bob trusts the root CA

An Example



Public Key Infrastructure (PKI)
CA
Issue
Check
User
User's
private key
Public key
certificate
Verifier

Using the PKI

Let $\mathcal{C}$ be Alice's certificate...

Alice		Bob
$\sigma := \mathsf{Sign}(sk, m)$		
	$\xrightarrow{\mathrm{m}, \sigma, \mathcal{C}}$	
		Extract pk from $\mathcal{C}$ and verify $\mathcal{C}$
		$b = \mathsf{Verify}(pk, m', \sigma)$
		Accept m' if $b = 1$

Bob verifies $\mathcal{C}$ by checking to see if it belongs to a certificate chain rooted at one of his trust anchors.

One Downside of PKI

Certificates are big!

Security Level	Public Key Size (bits)		Ratio ECC/RSA	Certificate Size (bits)	
	ECC	**RSA**		**ECDSA**	**RSA**
80	192	1024	5x smaller	577	2048
112	224	2048	9x smaller	673	4096
128	256	3072	12x smaller	769	6144
192	384	7680	20x smaller	1153	15360
256	521	15360	29x smaller	1564	30720

Implicit Certificates

Enter **Implicit Certificates**

Implicit Certificates

Enter **Implicit Certificates**

Idea: Public-keys are **derived** from certificates – neither a signature from the CA nor the entity's public key are explicitly included in the certificate

Implicit Certificates

Enter **Implicit Certificates**

Idea: Public-keys are **derived** from certificates – neither a signature from the CA nor the entity's public key are explicitly included in the certificate

Various advantages, including:

- Smaller (23x smaller than RSA certificates with 128-bit security)
- Faster (deriving a public key is faster than verifying a digital signature)
- *Less round trips*

How Much Smaller?

Table 2: Size comparison between ECC and RSA public key and certificates.

Security Level	Public key size[a] (bits)		Ratio ECC/RSA public keys	Certificate size[b] (bits)			Ratio ECQV/RSA certificates
	ECC	RSA		ECQV	ECDSA	RSA	
80	192	1024	5x smaller	193	577	2048	10x smaller
112	224	2048	9x smaller	225	673	4096	18x smaller
128	256	3072	12x smaller	257	769	6144	23x smaller
192	384	7680	20x smaller	385	1153	15360	39x smaller
256	521	15360	29x smaller	522	1564	30720	57x smaller

Data show that ECC certificates are 1-2 orders of magnitude smaller than RSA certificates, depending on security level. While ECDSA certificates are a factor 4-20 smaller than RSA certificates, ECQV implicit certificates realise another factor 3 of size reduction.

[a] *NIST-recommended key sizes.*
[b] *Data based on size of public keys and digital signatures, excluding (fixed) overhead of identification data.*

How Much Faster?

Table 3: Operational Cost Comparisons: *Conventional versus Implicit Certificates*

	Conventional Certificate		Implicit Certificate	
Action	Operation	Cost	Operation	Cost
1. Deriving the public key.	public key extraction (key included in cert)	**0**	compute public key from signature	**Elliptic-curve point multiplication**
2. Check authenticity of public keys (binding between the entity and the public key).	signature verification	**public-key operation**	no operation (delegated to Step 3)	**0**
3. Check authenticity of public keys in operation (binding between the entity and the private key).	evidenced by proper execution of protocol	**relatively expensive private-key operation** (as part of protocol)	evidenced by proper execution of protocol	**EC private-key operation (as part of protocol)**

Optimal Mail Certificates

$\underline{U}$		$\underline{CA}$
$r_U \xleftarrow{\$} [1, n-1]$ $R_U := k_U G$		
	$\xrightarrow{\mathrm{U}, \mathrm{R_U}}$	
		$k \xleftarrow{\$} [1, n-1]$ $P_U := R_U + kG$ $Cert_U := Encode(P_U, U, *)$ $e := H_n(Cert_U)$ $s := ed_{CA} + k \mod n$
	$\xleftarrow{\mathrm{r}, \mathrm{P_U}, \mathrm{Cert_U}}$	
$d_U := s + r_U \mod n$ $e := H_n(Cert_U)$ $Q_U := eQ_{CA} + P_U$		

Comments on Security

Both ECDSA and OMC are secure in isolation.
Similar security does not hold under composition [1].

[1] Brown, Daniel RL, Matthew J. Campagna, and Scott A. Vanstone. "Security of ECQV-Certified ECDSA Against Passive Adversaries." IACR Cryptology ePrint Archive 2009 (2009): 620.

The Attack Intuition

Goal: forge the signature on a message m without knowing U's private key

1. Let $r := X(fQ_{CA})$ (for any integer f, and where $X(\cdot)$ returns the x-coordinate of the point fQ_{CA})
2. $P_U := -H(M)r^{-1}G$ (the implicit certificate)
3. $s := rf^{-1}e \mod n$
4. The forged signature is (r, s), and the fake certificate is P

Verification

Bob knows the forged signature (r, s), M, U, the OMC certificate P_U, and the CA public key Q_{CA}. Compute:

$$
\begin{aligned}
Y &:= s^{-1}(H(M)G + rQ_U) \\
&:= s^{-1}(H(M)G) + s^{-1}r(eQ_{CA} + P_U)) \\
&:= s^{-1}(H(M)G) + s^{-1}reQ_{CA} - H(M)Gs^{-1}rr^{-1}) \\
&:= s^{-1}reQ_{CA} \\
&:= (rf^{-1}e)^{-1}reQ_{CA} \\
&:= fQ_{CA}
\end{aligned}
$$

This verifies since Y computed equals r provided

Trivial Avoidance

Solution: A verifier can check that $X(fQ_{CA})P = -H(M)G$ – if so, possible forgery.
Workaround Forgery: Select l and f from $[1, n-1]$ and do the following:

1. Let $r := X(fQ_{CA})$ (for any integer f, and where $X(\cdot)$ returns the x-coordinate of the point fQ_{CA})
2. $P_U := lQ_{CA} - H(M)r^{-1}G$ (the implicit certificate)
3. $s := (l + re)f^{-1} \mod n$
4. The forged signature is (r, s), and the fake certificate is P

Note: the first attack occurs when $l = 0$. We're generalizing here...

Workaround Check

$$
\begin{aligned}
Y &:= s^{-1}(H(M)G + rQ_U) \\
&:= s^{-1}(H(M)G + s^{-1}r(eQ_{CA} + P_U)) \\
&:= s^{-1}(H(M)G + r(lQ_{CA} - H(M)Gr^{-1} + eQ_{CA}) \\
&:= s^{-1}(rlQ_{CA} + reQ_{CA}) \\
&:= (f(rlQ_{CA} + reQ_{CA}))/(lr + er) \\
&:= (fQ_{CA}(rl + re))/(lr + er) \\
&:= fQ_{CA}
\end{aligned}
$$

Since $X(Y) = X(fQ_{CA}) = r$, we accept. This version is not detectable since all values of l consititute legitimate signatures.

Qa-Vanstone (ECQV) Implicit Certificate Scheme

The ECQV scheme is composed of six parts [1]:

1. Setup: Agree on all system parameters, e.g., (q, a, b, G), hash function, etc.
2. Certificate request: Generate a request for a certificate from the CA
3. Certificate generate: Verify the requestor's identity and create an implicit certificate
4. Certificate key extraction: Compute the public key from the implicit certificate
5. Certificate reception: Check the validity of the assigned public/private key pair

[1] SEC 4: Elliptic Curve Qu-Vanstone Implicit Certificate Scheme (ECQV), Certicom Research.

http://www.secg.org/sec4-1.0.pdf

ECQV Protocol

$\underline{U}$		$\underline{CA}$
$r_U \xleftarrow{\$} [1, n-1]$ $R_U := r_U G$		
	$\xrightarrow{\mathrm{U, R_U}}$	
		$k \xleftarrow{\$} [1, n-1]$ $P_U := R_U + kG$ $Cert_U := Encode(P_U, U, *)$ $e := H_n(Cert_U)$ $s := ek + d_{CA} \mod n$
	$\xleftarrow{\mathrm{r, Cert_U}}$	
$d_U := er_U + s \mod n$ $e := H_n(Cert_U)$ $Q_U := eP_U + Q_{CA}$		

Notable Differences

OMC and ECQV are close, but differ in one key way:

- OMC: $s = k + ed_{CA} \mod n$
- ECQV: $s = d_{CA} + ek$

The same attack does not hold!

Comments on Security

Theorem [1]: The ECQV implicit certificate scheme, when composed with ECDSA, is secure against passive adversaries under the combined assumption of the random oracle model and the generic group model.

- **Random Oracle**: An (adversarial-accessible) oracle in which every query is provided with a truly random and appropriate response, i.e., one chosen from the output domain. Previous queries are always supplied the same answer.
- **Generic Group**: The adversary's query is computed over elements of a generic group – i.e., not a finite field or elliptic curve.

[1] Brown, Daniel RL, Matthew J. Campagna, and Scott A. Vanstone. "Security of ECQV-Certified ECDSA Against Passive Adversaries." IACR Cryptology ePrint Archive 2009 (2009): 620.

Questions?