CSE 484 (Winter 2011)

Asymmetric Cryptography

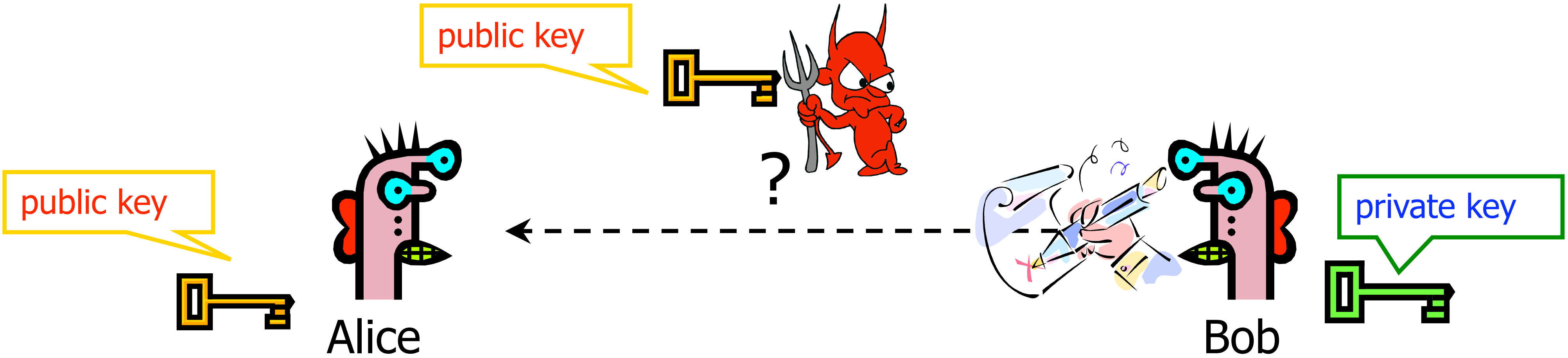
Tadayoshi Kohno

Thanks to Dan Boneh, Dieter Gollmann, John Manferdelli, John Mitchell, Vitaly Shmatikov, Bennet Yee, and many others for sample slides and materials ...

Goals for Today

- Asymmetric Cryptography
- HW2 returned at end of class (please remember to wait to pick up)

Digital Signatures: Basic Idea



<u>Given</u>: Everybody knows Bob's public key
Only Bob knows the corresponding private key

<u>Goal</u>: Bob sends a "digitally signed" message

1. To compute a signature, must know the private key
2. To verify a signature, enough to know the public key

RSA Signatures

- Public key is (n,e), private key is d
- To sign message m: $s = m^d \bmod n$
  - Signing and decryption are the same **underlying** operation in RSA
  - It's infeasible to compute s on m if you don't know d
- To verify signature s on message m:

  $s^e \bmod n = (m^d)^e \bmod n = m$
  - Just like encryption
  - Anyone who knows n and e (public key) can verify signatures produced with d (private key)
- In practice, also need padding & hashing
  - Standard padding/hashing schemes exist for RSA signatures

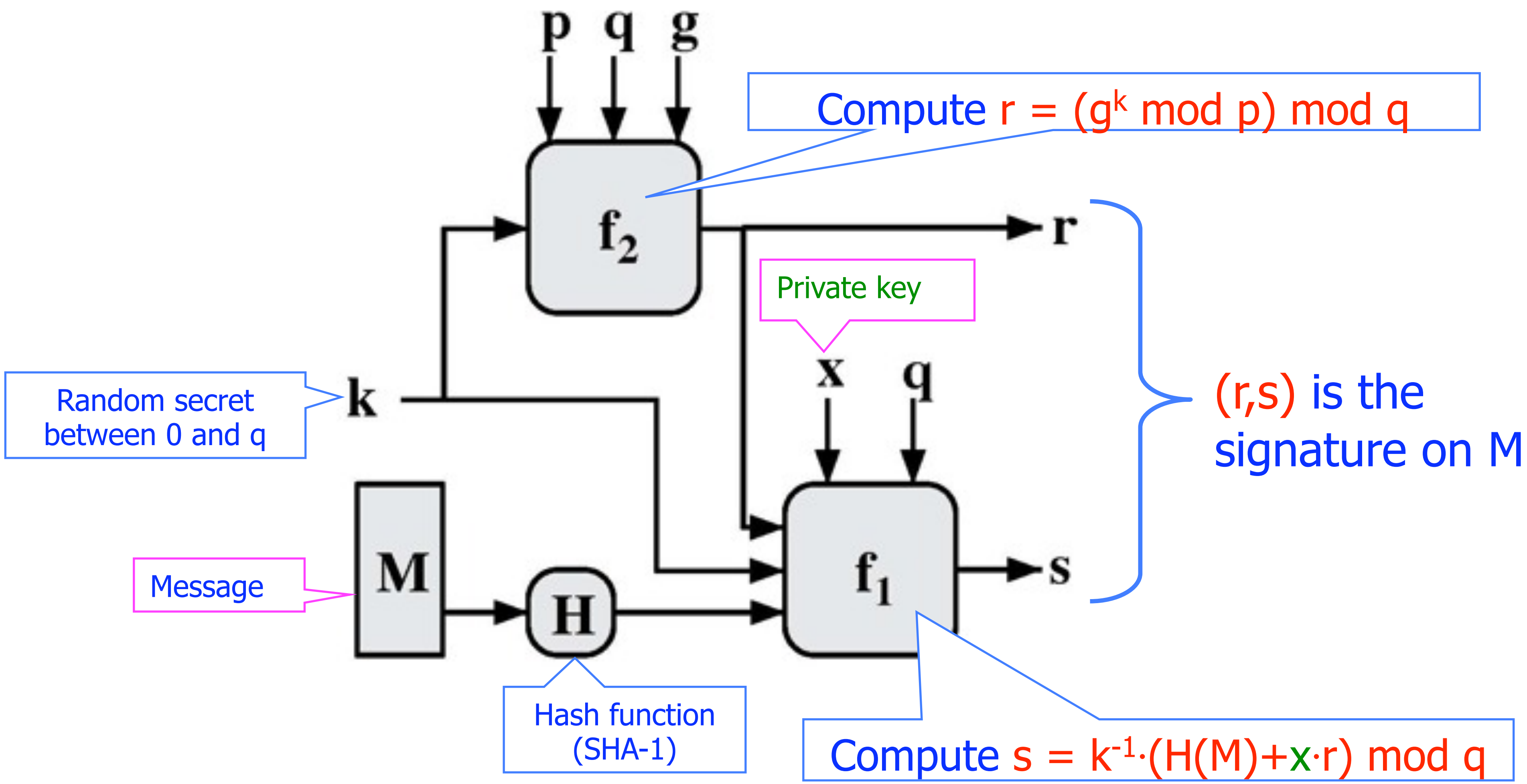
Encryption and Signatures

- Often people think: Encryption and decryption are inverses.
- That's a common view
  - True for the RSA **primitive (underlying component)**
- But not one we'll take
  - To really use RSA, we need padding
  - And there are many other decryption methods

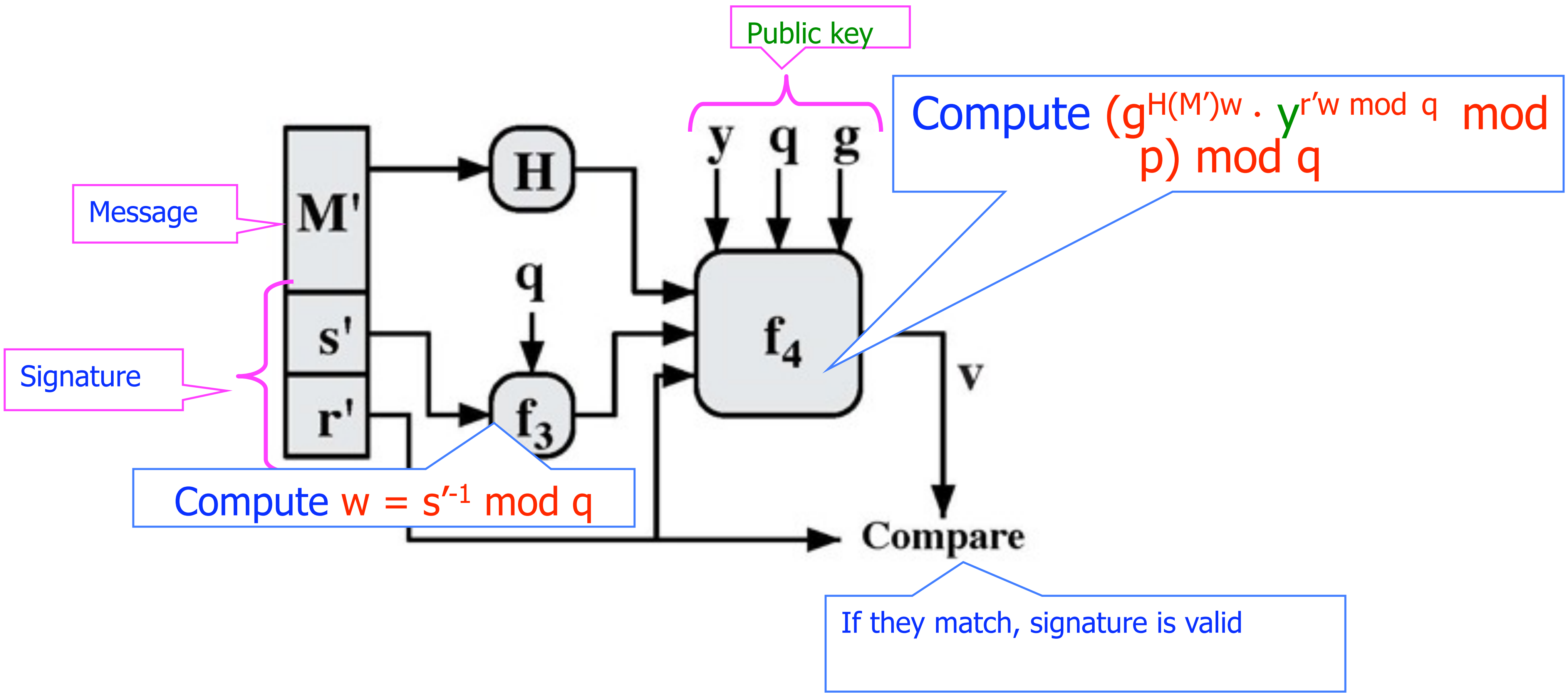
Digital Signature Standard (DSS)

- U.S. government standard (1991-94)
  - Modification of the ElGamal signature scheme (1985)
- Key generation:
  - Generate large primes p, q such that q divides p-1
    - $2^{159} < q < 2^{160}$, $2^{511+64t} < p < 2^{512+64t}$ where $0 \le t \le 8$
  - Select $h \in Z_p^*$ and compute $g = h^{(p-1)/q} \bmod p$
  - Select random x such $1 \le x \le q-1$, compute $y = g^x \bmod p$
- Public key: $(p, q, g, y = g^x \bmod p)$, private key: x
- Security of DSS requires hardness of discrete log
  - If could solve discrete logarithm problem, would extract x (private key) from $g^x \bmod p$ (public key)

DSS: Signing a Message (Skim)



DSS: Verifying a Signature (Skim)



Why DSS Verification Works (Skim)

- If (r,s) is a legitimate signature, then
  $r = (g^k \bmod p) \bmod q$; $s = k^{-1} \cdot (H(M)+x \cdot r) \bmod q$
- Thus $H(M) = -x \cdot r + k \cdot s \bmod q$
  - Multiply both sides by $w = s^{-1} \bmod q$
- $H(M) \cdot w + x \cdot r \cdot w = k \bmod q$
  - Exponentiate g to both sides
- $(g^{H(M) \cdot w + x \cdot r \cdot w} = g^k) \bmod p \bmod q$
  - In a valid signature, $g^k \bmod p \bmod q = r$, $g^x \bmod p = y$
- Verify $g^{H(M) \cdot w} \cdot y^{r \cdot w} = r \bmod p \bmod q$

Security of DSS

- Can't create a valid signature without private key
- Given a signature, hard to recover private key
- Can't change or tamper with signed message
- If the same message is signed twice, signatures are different
  - Each signature is based in part on random secret k
- Secret k must be different for each signature!
  - If k is leaked or if two messages re-use the same k, attacker can recover secret key x and forge any signature from then on
  - Example problem scenario: rebooted VMs; restarted embedded machines, Sony PS3!

Advantages of Public-Key Crypto

- Confidentiality without shared secrets
  - Very useful in open environments
  - No "chicken-and-egg" key establishment problem
    - With symmetric crypto, two parties must share a secret before they can exchange secret messages
    - Caveats to come
- Authentication without shared secrets
  - Use digital signatures to prove the origin of messages
- Reduce protection of information to protection of authenticity of public keys
  - No need to keep public keys secret, but must be sure that Alice's public key is <u>really</u> her true public key

Disadvantages of Public-Key Crypto

- Calculations are 2-3 orders of magnitude slower
  - Modular exponentiation is an expensive computation
  - Typical usage: use public-key cryptography to establish a shared secret, then switch to symmetric crypto
    - E.g., IPsec, SSL, SSH, ...
- Keys are longer
  - 1024+ bits (RSA) rather than 128 bits (AES)
- Relies on unproven number-theoretic assumptions
  - What if factoring is easy?
    - Factoring is believed to be neither P, nor NP-complete
  - (Of course, symmetric crypto also rests on unproven assumptions)

Exponentiation

- How to compute M^x mod N?
- Say, x = 13
- Sums of power of 2, x = 8+4+1 = $2^3+2^2+2^0$
- Can also write x in binary, e.g., x = 1101
- Can solve by repeated squaring
  - y = 1;
  - $y = y^2 * M \bmod N$ // $y = M$
  - $y = y^2 * M \bmod N$ // $y = M^2 * M = M^{2+1} = M^3$
  - $y = y^2 \bmod N$ // $y = (M^3)^2 = M^6$
  - $y = y^2 * M \bmod N$ // $y = (M^6)^2 * M = M^{12+1} = M^{13} = M^x$

Timing attacks

Collect timings for exponentiation with a bunch of messages M1, M2, ... (e.g., RSA signing operations with a private exponent)

Assume (inductively) know b_3=1, b_2=1, guess b_1=1

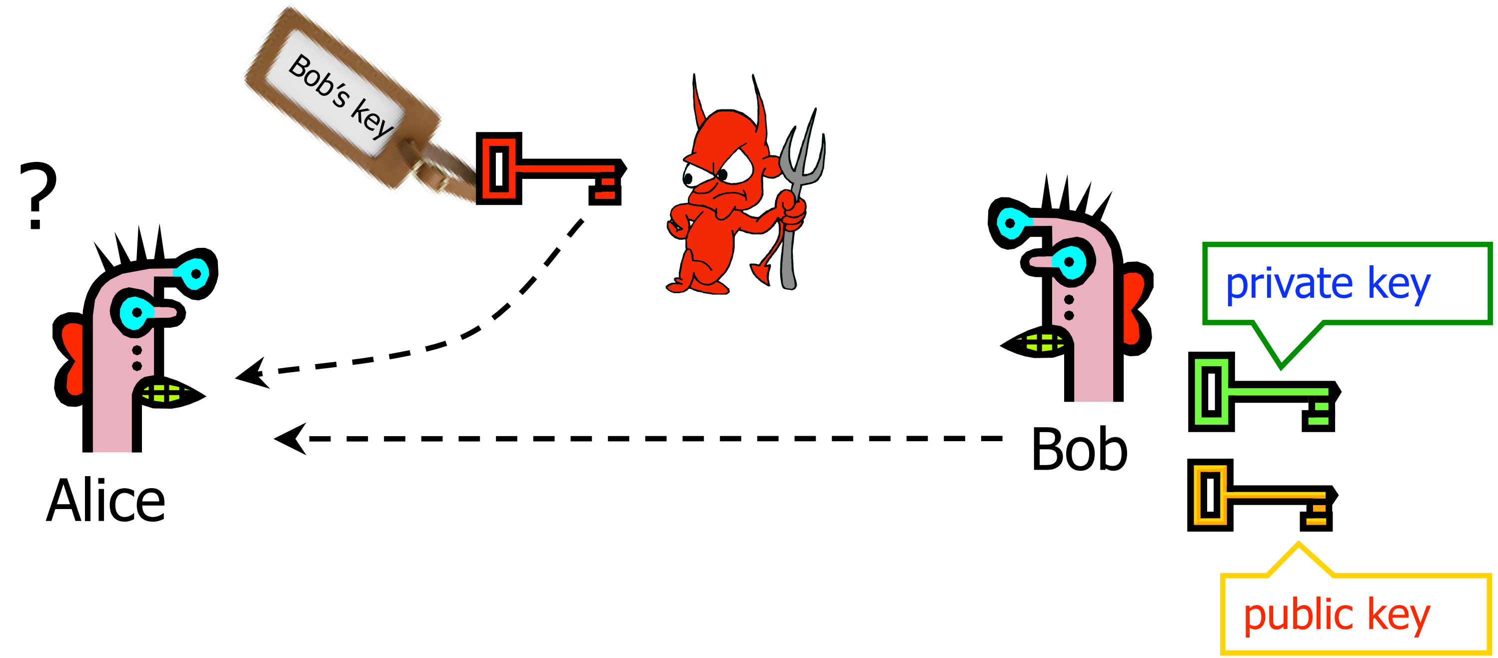
i	b_i = 0	b_i = 1	Comp	Meas
3	$y = y^2 \bmod N$	$y = y^2 * M1 \bmod N$		
2	$y = y^2 \bmod N$	$y = y^2 * M1 \bmod N$		
1	$y = y^2 \bmod N$	$y = y^2 * M1 \bmod N$	X1 secs	
0	$y = y^2 \bmod N$	$y = y^2 * M1 \bmod N$		Y1 secs

i	b_i = 0	b_i = 1	Comp	Meas
3	$y = y^2 \bmod N$	$y = y^2 * M2 \bmod N$		
2	$y = y^2 \bmod N$	$y = y^2 * M2 \bmod N$		
1	$y = y^2 \bmod N$	$y = y^2 * M2 \bmod N$	X2 secs	
0	$y = y^2 \bmod N$	$y = y^2 * M2 \bmod N$		Y2 secs

Timing attacks

- If $b_1 = 1$, then set of { Yj - Xj | j in {1,2, ..} } has distribution with "small" variance (due to time for final step, i=0)
  - "Guess" was correct when we computed X1, X2, ...
- If $b_1 = 0$, then set of { Yj - Xj | j in {1,2, ..} } has distribution with "large" variance (due to time for final step, i=0, and incorrect guess for b_1)
  - "Guess" was incorrect when we computed X1, X2, ...
  - So time computation wrong (Xj computed as large, but really small, ...)
- Strategy: Force user to sign large number of messages M1, M2, ... Record timings for signing.
- Iteratively learn bits of key by using above property.

Authenticity of Public Keys



Problem: How does Alice know that the public key she received is really Bob's public key?

Distribution of Public Keys

- Public announcement or public directory
  - Risks: forgery and tampering
- Public-key certificate
  - Signed statement specifying the key and identity
    - sig_{CA}("Bob", PK_B)
- Common approach: certificate authority (CA)
  - Single agency responsible for certifying public keys
  - After generating a private/public key pair, user proves his identity and knowledge of the private key to obtain CA's certificate for the public key (offline)
  - Every computer is pre-configured with CA's public key

Hierarchical Approach

- Single CA certifying every public key is impractical
- Instead, use a trusted root authority
  - For example, Verisign
  - Everybody must know the public key for verifying root authority's signatures
- Root authority signs certificates for lower-level authorities, lower-level authorities sign certificates for individual networks, and so on
  - Instead of a single certificate, use a certificate chain
    - $sig_{Verisign}$("AnotherCA", $PK_{AnotherCA}$), $sig_{AnotherCA}$("Alice", PK_A)
  - What happens if root authority is ever compromised?

Many Challenges

Spoofing URLs With Unicode

Posted by timothy **on Mon May 27, '02 09:48 PM**
from the **there-is-a-problem-with-this-certificate** dept.

Embedded Geek writes:

> "Scientific American has an interesting article about how a pair of students at the Technion-Israel Institute of Technology registered "microsoft.com" with Verisign, using the Russian Cyrillic letters "c" and "o". Even though it is a completely different domain, the two display identically (the article uses the term "homograph"). The work was done for a paper in the **Communications of the ACM** (the paper itself is not online). The article characterizes attacks using this spoof as "scary, if not entirely probable," assuming that a hacker would have to first take over a page at another site. I disagree: sending out a mail message with the URL waiting to be clicked ("Bill Gates will send you ten dollars!") is just one alternate technique. While security problems with Unicode have been noted here before, this might be a new twist."

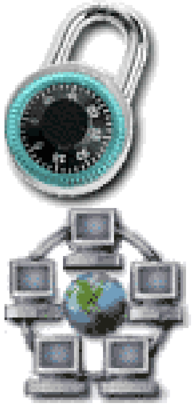
http://it.slashdot.org/story/08/12/30/1655234/CCC-Create-a-Rogue-CA-Certificate
http://www.win.tue.nl/hashclash/rogue-ca/

Many Challenges

CCC Create a Rogue CA Certificate

Posted by CmdrTaco **on Tue Dec 30, 2008 12:14 PM**
from the **they-even-faked-this-dept** dept.

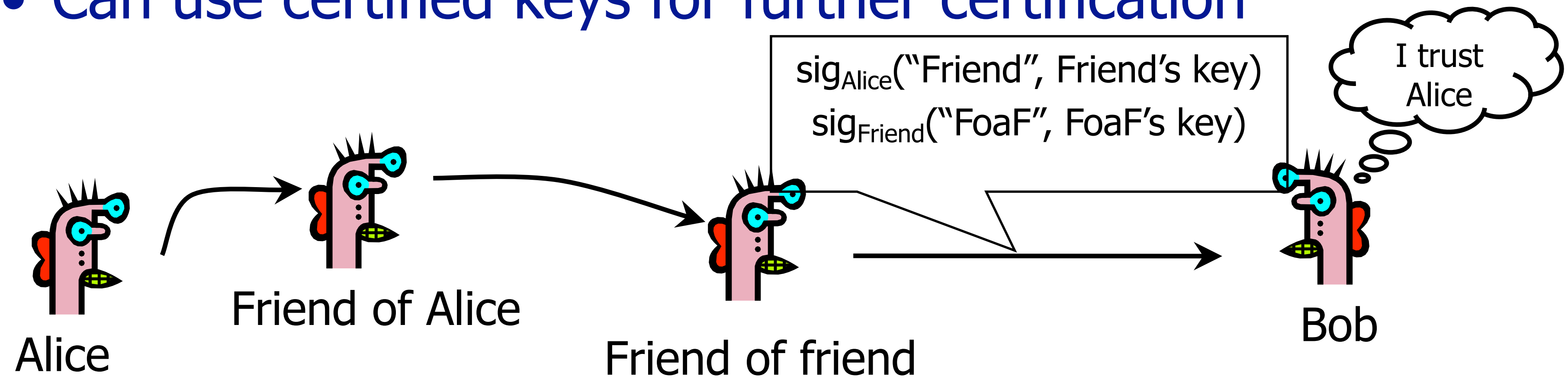
t3rmin4t0r writes



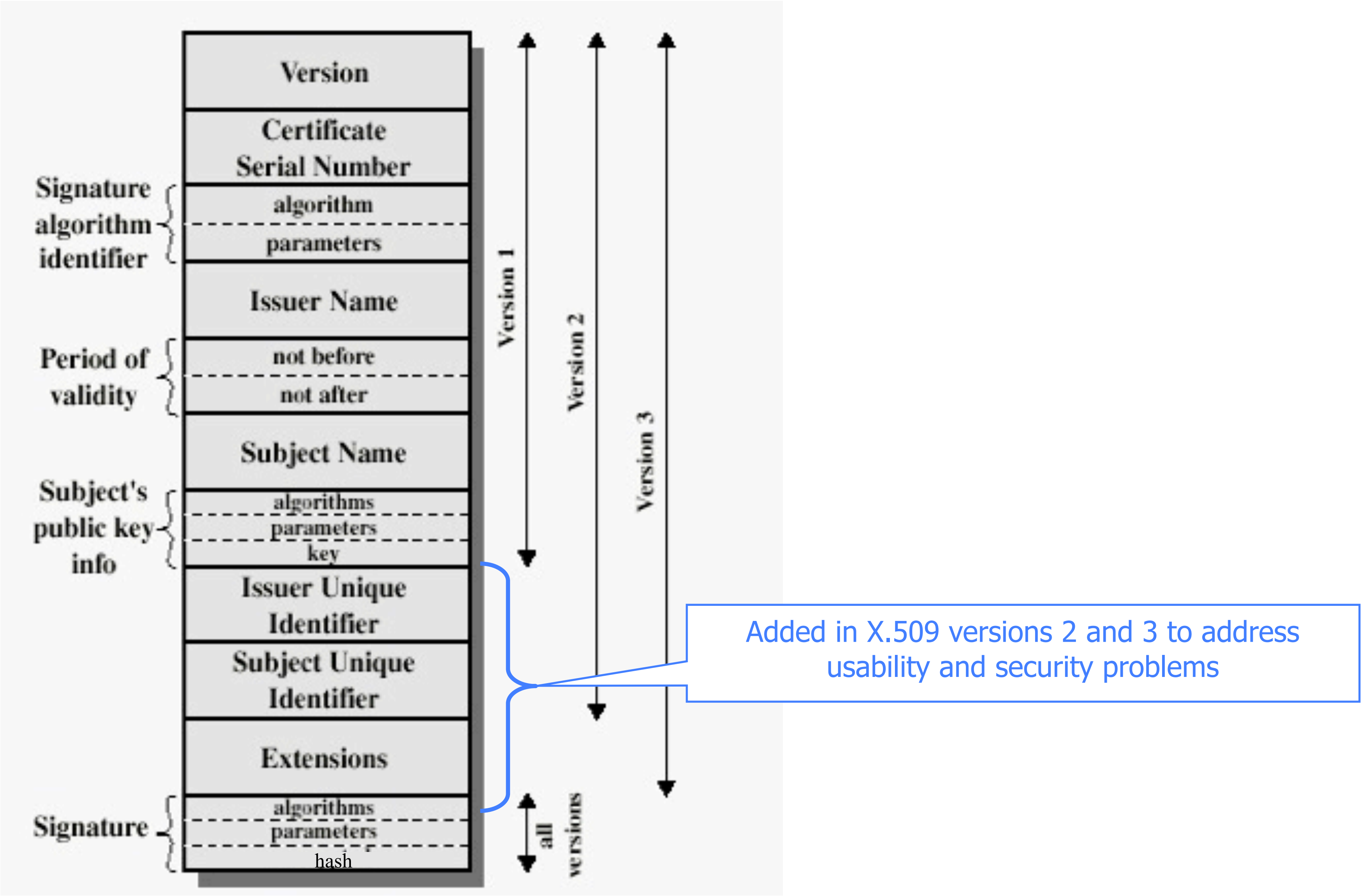
> "Just when you were breathing easy about Kaminsky, DNS and the word hijacking, by repeating the word SSL in your head, the hackers at CCC were busy at work making a hash of SSL certificate security. Here's the scoop on how they set up their own rogue CA, by (from what I can figure) reversing the hash and engineering a collision up in MD5 space. Until now, MD5 collisions have been ignored because nobody would put in that much effort to create a useful dummy file, but a CA certificate for phishing seems juicy enough to be fodder for the botnets now."

Alternative: "Web of Trust"

- Used in PGP (Pretty Good Privacy)
- Instead of a single root certificate authority, each person has a set of keys they "trust"
  - If public-key certificate is signed by one of the "trusted" keys, the public key contained in it will be deemed valid
- Trust can be transitive
  - Can use certified keys for further certification



X.509 Certificate



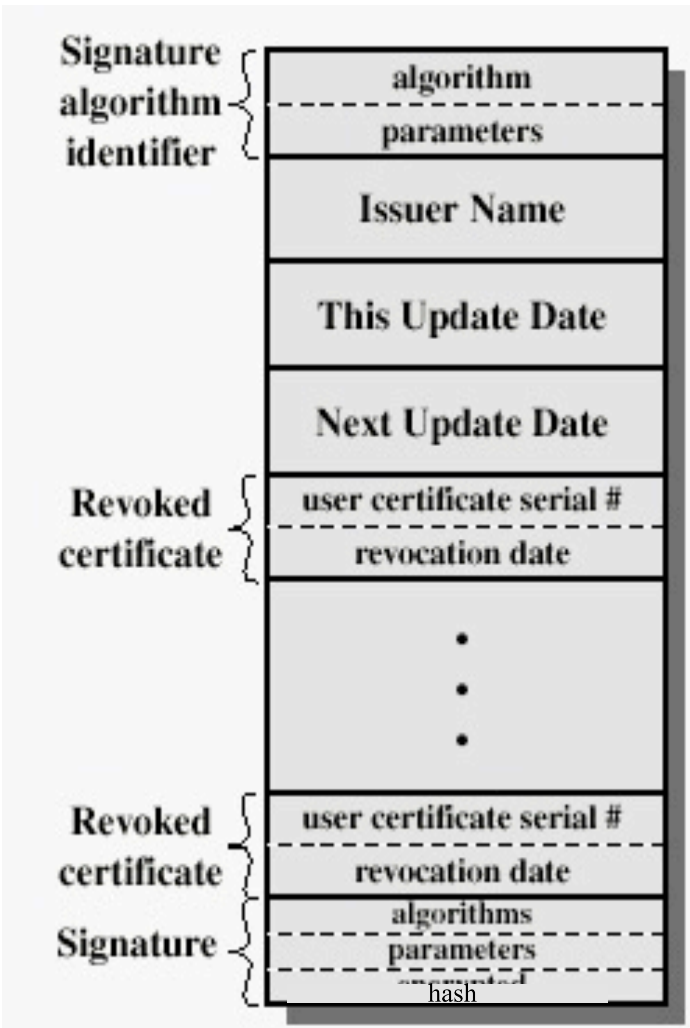
Certificate Revocation

- Revocation is <u>very</u> important
- Many valid reasons to revoke a certificate
  - Private key corresponding to the certified public key has been compromised
  - User stopped paying his certification fee to this CA and CA no longer wishes to certify him
  - CA's private key has been compromised!
- Expiration is a form of revocation, too
  - Many deployed systems don't bother with revocation
  - Re-issuance of certificates is a big revenue source for certificate authorities

Certificate Revocation Mechanisms

- Online revocation service
  - When a certificate is presented, recipient goes to a special online service to verify whether it is still valid
    - Like a merchant dialing up the credit card processor
- Certificate revocation list (CRL)
  - CA periodically issues a signed list of revoked certificates
    - Credit card companies used to issue thick books of canceled credit card numbers
  - Can issue a “delta CRL” containing only updates

X.509 Certificate Revocation List



Because certificate serial numbers must be unique within each CA, this is enough to identify the certificate